



Entwicklerhandbuch

Amazon API Gateway



Amazon API Gateway: Entwicklerhandbuch

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon API Gateway?	1
Architektur von API Gateway	2
Funktionen von API Gateway	3
Anwendungsfälle für API Gateway	3
Verwenden Sie API Gateway, um REST zu erstellen APIs	4
Verwenden Sie API Gateway, um HTTP zu erstellen APIs	5
Verwenden Sie API Gateway zum Erstellen WebSocket APIs	5
Von wem wird API Gateway verwendet?	6
Auf API Gateway zugreifen	7
Teil der AWS serverlosen Infrastruktur	7
So steigen Sie in Amazon API Gateway ein:	8
API Gateway-Konzepte	8
Wählen Sie zwischen REST APIs und HTTP APIs	14
Endpunkttyp	15
Sicherheit	15
Autorisierung	15
API-Management	16
Entwicklung	17
Überwachen	17
Integrationen	18
Erste Schritte mit der REST-API-Konsole	19
Schritt 1: Erstellen einer Lambda-Funktion	20
Schritt 2: Erstellen einer REST-API	21
Schritt 3: Erstellen einer Lambda-Proxy-Integration	21
Schritt 4: Bereitstellen Ihrer API	22
Schritt 5: Aufrufen Ihrer API	22
(Optional) Schritt 6: Bereinigen	23
Voraussetzungen	25
Melde dich an für ein AWS-Konto	25
Erste Schritte	26
Schritt 1: Erstellen einer Lambda-Funktion	27
Schritt 2: HTTP-API erstellen	27
Schritt 3: Ihre API testen	28
(Optional) Schritt 4: Bereinigen	29

Nächste Schritte	31
Tutorials und Workshops	32
REST API-Tutorials	33
Wählen Sie ein Tutorial zur Lambda-Integration aus	33
Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels	58
Wählen Sie ein Tutorial zur HTTP-Integration aus	68
Tutorial: Erstellen einer REST-API mit einer privaten Integration	84
Tutorial: Erstellen Sie eine REST-API mit einer AWS Integration	89
Tutorial: Erstellen einer Rechner-REST-API mit drei Integrationen	96
Tutorial: REST-API als AmazonS3-Proxy erstellen	124
Tutorial: REST-API als Amazon Kinesis-Proxy erstellen	169
Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI	217
Tutorial: Erstellen einer privaten REST-API	250
HTTP API-Tutorials	257
Tutorial: Entwickeln einer CRUD-HTTP-API mit Lambda und DynamoDB	257
Tutorial: Erstellen einer HTTP-API mit privater Integration für Amazon-ECS	270
WebSocket-API-Tutorials	277
Tutorial: Erstellen Sie eine Chat-App WebSocket	278
Tutorial: Erstellen Sie eine WebSocket API mit einer AWS Integration	283
API-Gateway-REST-APIs	298
Entwickeln	298
Beispielressource für eine REST-API	299
Zusätzliche REST-API-Features für die Entwicklung	302
API-Gateway-Endpunkttypen	304
Sicherheitsrichtlinien	308
IP-Adresstypen für REST-APIs in API Gateway	328
Methoden	330
Zugriffskontrolle	351
Integrationen	443
Anforderungvalidierung	539
Datentransformationen	570
Gateway-Antworten	654
CORES	667
Binäre Medientypen	682
Aufrufen	715
OpenAPI	749

Veröffentlichen	764
REST-APIs bereitstellen	765
Benutzerdefinierte Domänennamen	814
Optimieren	891
Cache-Einstellungen	891
Inhaltskodierung	905
Verteilen	911
Nutzungspläne	911
API-Dokumentation	939
SDK-Generierung	1007
Verkaufen Sie Ihre APIs als SaaS	1034
Schutz	1039
Gegenseitige TLS	1039
Clientzertifikate	1047
AWS WAF	1088
Drosselung	1091
Privates REST APIs	1094
Überwachen	1154
CloudWatch Metriken	1155
CloudWatch protokolliert	1165
Firehose	1171
Variablen für die Zugriffsprotokollierung in API Gateway	1173
X-Ray	1187
API-Gateway-Portale	1202
Überlegungen	1202
Portalprodukt	1203
Produktbeispiel für ein Portal zur Adoption von Haustieren	1203
Nächste Schritte	1205
Erstellen Sie ein Portalprodukt	1206
Erstellen Sie einen Produkt-REST-Endpunkt	1208
Erstellen Sie eine Produktseite	1210
Aktualisieren Sie ein Portalprodukt	1212
Aktualisieren Sie einen Produkt-REST-Endpunkt	1213
Produktseite aktualisieren	1214
Portalprodukte teilen	1214
Aktivieren Sie Try it für einen Produkt-REST-Endpunkt in Ihrem Portal	1220

Löschen Sie ein Portalprodukt	1222
Erstellen Sie ein Portal	1222
Wählen Sie einen Domainnamen	1223
Erstellen Sie ein Portal	1223
Aktualisieren Sie ein Portal	1226
Aktualisieren Sie ein Portal	1226
Eine Vorschau eines Portals anzeigen	1226
Überlegungen	1227
Eine Vorschau eines Portals anzeigen	1227
Veröffentlichen Sie ein Portal	1227
Überlegungen	1227
Veröffentlichen Sie ein Portal	1228
Verwenden Sie ein Portal	1228
Verwenden Sie ein Portal	1228
Beheben Sie Fehler in Ihrem Portal	1229
Deaktivieren Sie ein Portal	1230
Deaktivieren Sie ein Portal	1230
Löschen Sie ein Portal	1231
Löschen Sie ein Portal	1231
API-Gateway-HTTP-APIs	1232
Entwickeln	1232
Erstellen einer HTTP-API	1233
Routen	1234
IP-Adresstypen für HTTP-APIs in API Gateway	1237
Zugriffskontrolle	1239
Integrationen	1259
CORS	1278
Parameterzuweisung	1281
OpenAPI	1288
Veröffentlichen	1298
Stufen	1299
Sicherheitsrichtlinie für HTTP APIs in API Gateway	1302
Benutzerdefinierte Domainnamen	1304
Schutz	1314
Drosselung	1314
Gegenseitige TLS	1316

Überwachen	1322
Metriken	1323
Protokollierung	1326
Fehlerbehebung	1338
Lambda-Integrationen	1338
JWT-Genehmiger	1341
API-Gateway-WebSocket-API	1343
Überblick über WebSocket APIs	1343
Verwenden Sie Routen für eine API WebSocket	1344
Daten an verbundene Client-Apps senden	1345
WebSocket API-Statuscodes	1345
Verwalten von verbundenen Benutzern und Client-Apps	1346
Aufrufen Ihrer Backend-Integration	1350
WebSocket Auswahlausdrücke	1354
Entwickeln	1365
Erstellen und Konfigurieren	1365
IP-Adresstypen für WebSocket-APIs im API Gateway	1367
Routen	1369
Zugriffskontrolle	1378
Integrationen	1387
Anforderungvalidierung	1397
Datentransformationen	1401
Binäre Medientypen	1414
Invoke	1414
Veröffentlichen	1418
Stufen	1419
Stellen Sie eine WebSocket API bereit	1421
Sicherheitsrichtlinie für WebSocket APIs in API Gateway	1424
Benutzerdefinierte Domainnamen	1426
Schutz	1434
Drosselung auf Kontoebene pro Region	1435
Drosselung auf Routenebene	1435
Überwachen	1436
Kennzahlen	1436
Protokollierung	1439
API Gateway ARNs	1448

HTTP-API und WebSocket API-Ressourcen	1448
REST-API-Ressourcen	1451
Ressourcen für Domainnamen	1455
execute-api(HTTP APIs WebSocket APIs, und REST APIs)	1456
Erweiterungen für OpenAPI	1458
x-amazon-apigateway-any-method	1459
x-amazon-apigateway-any-Methodenbeispiele	1460
x-amazon-apigateway-cors	1461
x-amazon-apigateway-cors Beispiel	1461
x-amazon-apigateway-api-key-source	1463
x-amazon-apigateway-apiBeispiel für -key-source	1463
x-amazon-apigateway-auth	1464
x-amazon-apigateway-auth Beispiel	1464
x-amazon-apigateway-authorizer	1465
x-amazon-apigateway-authorizer Beispiele für REST APIs	1469
x-amazon-apigateway-authorizer Beispiele für HTTP APIs	1473
x-amazon-apigateway-authtype	1474
x-amazon-apigateway-authtype Beispiel	1475
Weitere Informationen finden Sie auch unter	1477
x-amazon-apigateway-binary-Medientyp	1477
x-amazon-apigateway-binaryBeispiel für -Medientypen	1477
x-amazon-apigateway-documentation	1477
x-amazon-apigateway-documentation Beispiel	1478
x-amazon-apigateway-endpoint-Zugriffsmodus	1478
x-amazon-apigateway-endpoint-access-modeBeispiel für	1478
x-amazon-apigateway-endpoint-Konfiguration	1479
x-amazon-apigateway-endpoint-Konfigurationsbeispiele	1480
x-amazon-apigateway-gateway-Antworten	1480
x-amazon-apigateway-gatewayBeispiel für -Antworten	1481
x-amazon-apigateway-gateway-Responses.GatewayResponse	1481
x-amazon-apigateway-gateway-Responses.GatewayResponse-Beispiel	1482
x-amazon-apigateway-gateway-Responses.ResponseParameters	1483
x-amazon-apigateway-gateway-Responses.ResponseParameters — Beispiel	1483
x-amazon-apigateway-gateway-Responses.ResponseTemplates	1483
x-amazon-apigateway-gateway-Responses.ResponseTemplates — Beispiel	1484
x-amazon-apigateway-importexport-Ausführung	1485

x-amazon-apigateway-importexportBeispiel für eine Version	1485
x-amazon-apigateway-integration	1485
x-amazon-apigateway-integration Beispiele	1493
x-amazon-apigateway-integrations	1495
x-amazon-apigateway-integrations Beispiel	1495
x-amazon-apigateway-integration.requestTemplates	1497
x-amazon-apigateway-integration.requestTemplates Beispiel	1497
x-amazon-apigateway-integration. Anforderungsparameter	1498
x-amazon-apigateway-integration.requestParametersBeispiel für	1499
x-amazon-apigateway-integration. Antworten	1500
x-amazon-apigateway-integration.responsesBeispiel für	1501
x-amazon-apigateway-integration.response	1502
x-amazon-apigateway-integration.responseBeispiel für	1503
x-amazon-apigateway-integration. responseTemplates	1504
x-amazon-apigateway-integration.responseTemplate-Beispiel	1504
x-amazon-apigateway-integration. responseParameters	1505
x-amazon-apigateway-integration.responseParametersBeispiel für	1505
x-amazon-apigateway-integration.tlsConfig	1505
x-amazon-apigateway-integration.tlsConfig-Beispiele	1508
x-amazon-apigateway-minimum-Kompressionsgröße	1508
x-amazon-apigateway-minimumBeispiel für eine Kompressionsgröße	1509
x-amazon-apigateway-policy	1509
x-amazon-apigateway-policyBeispiel für	1509
x-amazon-apigateway-request-validator	1510
x-amazon-apigateway-request-validatorBeispiel für	1510
x-amazon-apigateway-request-Validatoren	1511
x-amazon-apigateway-request-validatorsBeispiel für	1512
x-amazon-apigateway-request-Validators.RequestValidator	1512
x-amazon-apigateway-request-validators.requestValidatorBeispiel für	1513
x-amazon-apigateway-security-Politik	1513
x-amazon-apigateway-security-policyBeispiel für	1513
x-amazon-apigateway-tag-Wert	1513
x-amazon-apigateway-tag-valueBeispiel für	1514
Sicherheit	1515
Datenschutz	1516
Datenverschlüsselung	1517

Richtlinie für den Datenverkehr zwischen Netzwerken	1519
Identitäts- und Zugriffsverwaltung	1520
Zielgruppe	1520
Authentifizierung mit Identitäten	1520
Verwalten des Zugriffs mit Richtlinien	1522
Funktionsweise von Amazon API Gateway mit IAM	1524
Beispiele für identitätsbasierte Richtlinien	1529
Beispiele für eine ressourcenbasierte Richtlinie	1542
Fehlersuche	1542
Verwenden von serviceverknüpften Rollen	1544
Protokollierung und Überwachung	1549
Arbeiten mit CloudTrail	1551
Arbeiten mit AWS Config	1554
Compliance-Validierung	1558
Ausfallsicherheit	1558
Sicherheit der Infrastruktur	1559
Konfigurations- und Schwachstellenanalyse	1559
Bewährte Methoden	1560
Markieren	1562
API Gateway-Ressourcen, die mit Tags versehen werden können	1563
Tag-Vererbung in der Amazon API Gateway V1-API	1564
Tag-Einschränkungen und Nutzungskonventionen	1565
Attributbasierte Zugriffskontrolle	1566
Einschränken von Aktionen basierend auf Ressourcen-Tags	1566
Zulassen von Aktionen basierend auf Ressourcen-Tags	1567
Tagging-Operationen verweigern	1568
Tagging-Operationen erlauben	1569
API-Referenzen	1571
Service-Endpunkte in API Gateway	1571
IPv4 Endpunkte	1571
Dualstack IPv4 - (und) Endpunkte IPv6	1572
Angaben von Endpunkten	1572
IPv6 Adressen in IAM-Richtlinien verwenden	1573
Kontingente und wichtige Hinweise	1575
API Gateway-Kontingent auf Kontoebene, pro Region	1575
API Gateway-Kontingente für das Erstellen, Bereitstellen und Verwalten einer API	1576

REST-API-Kontingente	1580
Bewährte Methoden zur Reduzierung Ihrer Kontingente	1586
HTTP-API-Kontingente	1588
Bewährte Methoden zur Reduzierung Ihrer Kontingente	1591
Portal-Kontingente	1591
WebSocket API-Kontingente	1592
Wichtige Hinweise	1595
Amazon API Gateway: Wichtige Hinweise für HTTP APIs	1596
Amazon API Gateway wichtige Hinweise für HTTP und WebSocket APIs	1596
Wichtige Hinweise für REST APIs und WebSocket APIs	1596
Wichtige Hinweise für WebSocket APIs	1597
Wichtige Hinweise für REST APIs	1597
Dokumentverlauf	1604
Frühere Updates	1619
.....	mdcxxxi

Was ist Amazon API Gateway?

Amazon API Gateway ist ein AWS Service für die Erstellung, Veröffentlichung, Wartung, Überwachung und Sicherung von REST, HTTP und WebSocket APIs in jeder Größenordnung. API-Entwickler können APIs diesen Zugriff AWS oder andere Webdienste sowie in der [AWS Cloud](#) gespeicherte Daten erstellen. Als API Gateway API-Entwickler können Sie API-Anwendungen APIs für die Verwendung in Ihren eigenen Client-Anwendungen erstellen. Oder Sie können Ihre App Entwicklern von Drittanbietern APIs zur Verfügung stellen. Weitere Informationen finden Sie unter [the section called “Von wem wird API Gateway verwendet?”](#).

API Gateway erstellt RESTful APIs das:

- HTTP-basiert sind
- Die zustandslose Client-Server-Kommunikation aktivieren.
- HTTP-Standardmethoden wie GET, POST, PUT, PATCH und DELETE implementieren.

Weitere Informationen zu API Gateway REST APIs und HTTP finden Sie unter APIs [the section called “Wählen Sie zwischen REST APIs und HTTP APIs ”API-Gateway-HTTP-APIs](#), [the section called “Verwenden Sie API Gateway, um REST zu erstellen APIs”](#), und [the section called “Entwickeln”](#).

API Gateway erstellt WebSocket APIs das:

- Halten Sie sich an das [WebSocket](#) Protokoll, das eine statusbehaftete Vollduplex-Kommunikation zwischen Client und Server ermöglicht.
- eingehende Nachrichten basierend auf dem Inhalt der Nachricht weiterleiten.

Weitere Informationen zu API Gateway WebSocket APIs finden Sie unter [the section called “Verwenden Sie API Gateway zum Erstellen WebSocket APIs”](#) und [the section called “Überblick über WebSocket APIs”](#).

Themen

- [Architektur von API Gateway](#)
- [Funktionen von API Gateway](#)
- [Anwendungsfälle für API Gateway](#)

- [Auf API Gateway zugreifen](#)
- [Teil der AWS serverlosen Infrastruktur](#)
- [So steigen Sie in Amazon API Gateway ein:](#)
- [Amazon API Gateway-Konzepte](#)
- [Wählen Sie zwischen REST APIs und HTTP APIs](#)
- [Erste Schritte mit der REST-API-Konsole](#)

Architektur von API Gateway

Das folgende Diagramm zeigt die API Gateway-Architektur.



Dieses Diagramm zeigt, wie das, APIs was Sie in Amazon API Gateway erstellen, Ihnen oder Ihren Entwicklerkunden ein integriertes und konsistentes Entwicklererlebnis für die Erstellung AWS serverloser Anwendungen bietet. API Gateway handhabt sämtliche Aufgaben im Zusammenhang mit der Annahme und Verarbeitung von Hunderttausenden gleichzeitiger API-Aufrufe. Zu diesen Aufgaben gehören die Verwaltung des Datenverkehrs, Autorisierung und Zugriffskontrolle, Überwachung und Verwaltung der API-Version.

API Gateway fungiert als „Eingangstür“ für Anwendungen, die auf Daten, Geschäftslogik oder Funktionen aus Ihren Backend-Services zugreifen können, z. B. Workloads, die auf Amazon Elastic Compute Cloud (Amazon EC2) ausgeführt werden, Code, der auf beliebigen Webanwendungen ausgeführt wird AWS Lambda, oder Echtzeitkommunikationsanwendungen.

Funktionen von API Gateway

Amazon API Gateway bietet zum Beispiel die folgenden Funktionen:

- Support für stateful ([WebSocket](#)) und stateless ([HTTP](#) und [REST](#)). APIs
- Leistungsstarke, flexible [Authentifizierungsmechanismen](#) wie AWS Identity and Access Management Richtlinien, Lambda-Autorisierungsfunktionen und Amazon Cognito Cognito-Benutzerpools.
- [Canary-Release-Bereitstellungen](#) für sichere fortlaufende Änderungen
- [CloudTrail](#)-Protokollierung und -Überwachung von API-Nutzung und API-Änderungen
- CloudWatch Zugriffs- und Ausführungsprotokollierung, einschließlich der Möglichkeit, Alarme einzustellen. Weitere Informationen erhalten Sie unter [the section called “CloudWatch Metriken”](#) und [the section called “Kennzahlen”](#).
- Möglichkeit, CloudFormation Vorlagen zu verwenden, um die API-Erstellung zu ermöglichen. Weitere Informationen finden Sie unter [Referenz der Amazon API Gateway-Ressourcentypen](#) und [Referenz der Amazon API Gateway V2-Ressourcentypen](#).
- Unterstützung [benutzerdefinierter Domännennamen](#)
- Integration mit [AWS WAF](#) zum Schutz APIs vor gängigen Web-Exploits.
- [AWS X-Ray](#)-Integration für ein besseres Verständnis und die Analyse leistungsbezogener Latenzen

Eine vollständige Liste der API Gateway-Funktionsversionen finden Sie unter [Dokumentverlauf](#).

Anwendungsfälle für API Gateway

Der folgende Abschnitt mit Anwendungsfällen bietet einen Überblick über die verschiedenen Arten von API-Gateways APIs und die verschiedenen Arten von Entwicklern, die API Gateway verwenden. Ausführlichere Informationen zum Unterschied zwischen REST APIs und HTTP APIs finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

Themen

- [Verwenden Sie API Gateway, um REST zu erstellen APIs](#)
- [Verwenden Sie API Gateway, um HTTP zu erstellen APIs](#)
- [Verwenden Sie API Gateway zum Erstellen WebSocket APIs](#)

- [Von wem wird API Gateway verwendet?](#)

Verwenden Sie API Gateway, um REST zu erstellen APIs

Eine API Gateway-REST-API setzt sich aus Ressourcen und Methoden zusammen. Eine Ressource ist eine logische Entität, auf die eine App über einen Ressourcenpfad zugreifen kann. Eine Methode entspricht einer REST-API-Anforderung, die vom API-Benutzer übermittelt wird, und der Antwort, die an den Benutzer zurückgegeben wird.

So könnte beispielsweise `/incomes` der Pfad der Ressource sein, die das Einkommen des App-Benutzers angibt. Eine Ressource kann für eine oder mehrere Operationen verwendet werden, die durch geeignete HTTP-Verben wie GET, POST, PUT, PATCH und DELETE definiert sind. Eine Kombination aus einem Ressourcenpfad und einer Operation identifiziert eine Methode der API. Die Methode POST `/incomes` beispielsweise könnte das Einkommen des Aufrufers hinzufügen, und die Methode GET `/expenses` könnte die vom Aufrufer angegebenen Ausgaben abfragen.

Die App benötigt keine Informationen dazu, wo auf dem Backend die angeforderten Daten gespeichert und abgerufen werden. In API Gateway REST APIs ist das Frontend durch Methodenanfragen und Methodenantworten gekapselt. Die API arbeitet über Integrationsanforderungen und Integrationsantworten mit dem Backend zusammen.

Wenn beispielsweise DynamoDB das Backend bildet, richtet der API-Entwickler die Integrationsanforderung so ein, dass die eingehende Methodenanforderung an das gewählte Backend weitergeleitet wird. Die Konfiguration umfasst Spezifikationen für eine geeignete DynamoDB-Aktion, eine erforderliche IAM-Rolle sowie -Richtlinien und die erforderlichen Eingabedaten. Das Backend gibt das Ergebnis als Integrationsantwort an API Gateway zurück.

Um die Integrationsantwort auf eine geeignete Methodenantwort (eines HTTP-Statuscodes) an den Client weiterzuleiten, können Sie die Integrationsantwort so konfigurieren, dass die erforderlichen Antwortparameter von der Integration der Methode zugewiesen werden. Falls erforderlich, übertragen Sie anschließend das Backend-Ausgabedatenformat in das für das Frontend erforderliche Format. API Gateway bietet die Möglichkeit, ein Schema oder Modell für die [Nutzlast](#) zu definieren, um die Einrichtung der Textzuweisungsvorlage zu erleichtern.

API Gateway bietet REST-API-Verwaltungsfunktionen wie die folgenden:

- Support für die Generierung SDKs und Erstellung von API-Dokumentation mithilfe von API Gateway Gateway-Erweiterungen für OpenAPI

- Drosselung von HTTP-Anforderungen

Verwenden Sie API Gateway, um HTTP zu erstellen APIs

HTTP APIs ermöglicht es Ihnen, RESTful APIs mit geringerer Latenz und geringeren Kosten als REST zu erstellen APIs.

Sie können HTTP verwenden APIs , um Anfragen an AWS Lambda Funktionen oder an jeden öffentlich routbaren HTTP-Endpunkt zu senden.

Beispielsweise können Sie eine HTTP-API erstellen, die in eine Lambda-Funktion auf dem Backend integriert wird. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion und gibt die Antwort der Funktion an den Client zurück.

APIs HTTP-Unterstützung [OpenID Connect](#) und [OAuth 2.0-Autorisierung](#). Sie verfügen über integrierten Support für Cross-Origin Resource Sharing (CORS) und automatische Bereitstellungen.

Weitere Informationen hierzu finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

Verwenden Sie API Gateway zum Erstellen WebSocket APIs

In einer WebSocket API können sowohl der Client als auch der Server jederzeit Nachrichten aneinander senden. Backend-Server können problemlos Daten an verbundene Benutzer und Geräte übertragen, ohne komplexe Abfragemechanismen implementieren zu müssen.

Sie könnten beispielsweise mithilfe einer WebSocket API Gateway eine serverlose Anwendung erstellen und Nachrichten AWS Lambda an und von einzelnen Benutzern oder Benutzergruppen in einem Chatroom senden und empfangen. Oder Sie könnten Back-End-Dienste wie AWS Lambda Amazon Kinesis oder einen HTTP-Endpunkt aufrufen, der auf Nachrichteninhalten basiert.

Sie können API Gateway verwenden, WebSocket APIs um sichere Kommunikationsanwendungen in Echtzeit zu erstellen, ohne Server zur Verwaltung von Verbindungen oder umfangreichem Datenaustausch bereitstellen oder verwalten zu müssen. Zu vorgesehenen Anwendungsfällen gehören Echtzeit-Anwendungen wie die folgenden:

- Chat-Anwendungen
- Echtzeit-Dashboards, wie z. B. Aktienticker

- Echtzeit-Warnungen und -Benachrichtigungen

API Gateway bietet WebSocket API-Verwaltungsfunktionen wie die folgenden:

- Überwachung und Drosselung von Verbindungen und Nachrichten
- Wird verwendet AWS X-Ray , um Nachrichten auf ihrem Weg durch die APIs Backend-Dienste zu verfolgen
- Einfache Integration mit Endpunkten HTTP/HTTPS

Von wem wird API Gateway verwendet?

Es gibt zwei Arten von Entwicklern, die API Gateway verwenden: API-Entwickler und Anwendungsentwickler.

Ein API-Entwickler erstellt eine API und stellt sie bereit, um die erforderliche Funktionalität in API Gateway zu aktivieren. Der API-Entwickler muss ein Benutzer des AWS Kontos sein, dem die API gehört.

Ein App-Entwickler erstellt eine funktionierende Anwendung zum Aufrufen von AWS Diensten, indem er eine WebSocket oder eine von einem API-Entwickler erstellte REST-API in API Gateway aufruft.

Der App-Entwickler ist der Kunde des API-Entwicklers. Der App-Entwickler benötigt kein AWS Konto, vorausgesetzt, die API benötigt entweder keine IAM-Berechtigungen oder unterstützt die Autorisierung von Benutzern über externe Federated Identity Provider, die von [Amazon Cognito User Pool](#) Identity Federation unterstützt werden. Zu diesen Identitätsanbietern gehören Amazon, Amazon Cognito-Benutzerpools, Facebook und Google.

API Gateway-API erstellen und verwalten

Ein API-Entwickler arbeitet mit der API Gateway-Service-Komponente für die API-Verwaltung (apigateway), um eine API zu erstellen, zu konfigurieren und bereitzustellen.

Als API-Entwickler können Sie eine API erstellen und verwalten, indem Sie die unter [Erste Schritte mit API-Gateway](#) beschriebene API Gateway-Konsole verwenden oder die [API-Referenzen](#) aufrufen. Es gibt mehrere Möglichkeiten, diese API aufzurufen. Dazu gehören die Verwendung von AWS Command Line Interface (AWS CLI) oder die Verwendung eines SDK. AWS Darüber hinaus können Sie die API-Erstellung mit [AWS CloudFormation Vorlagen](#) oder (im Fall von REST APIs und HTTP APIs) aktivieren [OpenAPI-Erweiterungen für API Gateway](#).

Eine Liste der Regionen, in denen API Gateway verfügbar ist, sowie der zugehörigen Kontrolldienstendpunkte finden Sie unter [Amazon API Gateway-Endpunkte und -Kontingente](#).

API Gateway-API aufrufen

Ein Anwendungsentwickler arbeitet mit der API Gateway Service-Komponente für die API-Ausführung (`execute-api`), um eine API aufzurufen, die in API Gateway erstellt oder bereitgestellt wurde. Die zugrunde liegenden Programmierentitäten werden von der erstellten API bereitgestellt. Es gibt mehrere Methoden für den Aufruf einer solchen API. Weitere Informationen hierzu finden Sie unter [Aufrufen von REST-APIs in API Gateway](#) und [Aufrufen WebSocket APIs](#).

Auf API Gateway zugreifen

Sie können wie folgt auf Amazon API Gateway zugreifen:

- **AWS-Managementkonsole**— Die AWS-Managementkonsole bietet eine Weboberfläche zum Erstellen und Verwalten APIs. Nachdem Sie die Schritte unter [Voraussetzungen](#) ausgeführt haben, können Sie unter <https://console.aws.amazon.com/apigateway> auf die API-Gateway-Konsole zugreifen.
- **AWS SDKs**— Wenn Sie eine Programmiersprache verwenden, die ein SDK für AWS bereitstellt, können Sie ein SDK für den Zugriff auf API Gateway verwenden. SDKs Vereinfachen Sie die Authentifizierung, lassen Sie sich problemlos in Ihre Entwicklungsumgebung integrieren und bieten Sie Zugriff auf API Gateway Gateway-Befehle. Weitere Informationen finden Sie unter [Tools für Amazon Web Services](#).
- **API Gateway V1 und V2 APIs** — Wenn Sie eine Programmiersprache verwenden, für die kein SDK verfügbar ist, lesen Sie die [Amazon API Gateway Version 1 API-Referenz](#) und [Amazon API Gateway Version 2 API-Referenz](#).
- **AWS Command Line Interface** – Weitere Informationen finden Sie unter [Einrichtung der AWS Command Line Interface](#) im AWS Command Line Interface -Benutzerhandbuch.
- **AWS Tools for Windows PowerShell** – Weitere Informationen finden Sie unter [Einrichten von AWS Tools for Windows PowerShell](#) im AWS -Tools für PowerShell Benutzerhandbuch.

Teil der AWS serverlosen Infrastruktur

Zusammen mit [AWS Lambda](#) API Gateway bildet das API Gateway den anwendungsorientierten Teil der AWS serverlosen Infrastruktur. Weitere Informationen zu den ersten Schritten mit Serverless-Technologie finden Sie im [Serverless-Entwicklerhandbuch](#).

Damit eine App öffentlich verfügbare AWS Dienste aufruft, können Sie Lambda verwenden, um mit den erforderlichen Diensten zu interagieren und Lambda-Funktionen über API-Methoden in API Gateway verfügbar zu machen. AWS Lambda führt Ihren Code auf einer hochverfügbaren Computerinfrastruktur aus. Es übernimmt die erforderliche Ausführung und Verwaltung der Datenverarbeitungsressourcen. Um serverlose Anwendungen zu ermöglichen, unterstützt API Gateway [optimierte Proxyintegrationen](#) mit AWS Lambda und HTTP-Endpunkten.

So steigen Sie in Amazon API Gateway ein:

Eine Einführung in Amazon API Gateway finden Sie in den folgenden Themen:

- [Erste Schritte](#), worin Sie eine Anleitung zum Erstellen einer HTTP-API erhalten.
- [Serverless Land](#), das Lehrvideos bietet.
- [Happy Little API Shorts](#), eine Reihe kurzer Lehrvideos.

Amazon API Gateway-Konzepte

Im folgenden Abschnitt werden einführende Konzepte für die Verwendung von API Gateway beschrieben.

API Gateway

API Gateway ist ein AWS Dienst, der Folgendes unterstützt:

- Erstellen, Bereitstellen und Verwalten einer [RESTful](#)Anwendungsprogrammierschnittstelle (API) zur Bereitstellung von HTTP-Endpunkten, AWS Lambda -Funktionen oder anderen AWS Diensten im Backend.
- Erstellen, Bereitstellen und Verwalten einer [WebSocket](#)API zur Bereitstellung von AWS Lambda Funktionen oder anderen AWS Diensten.
- Aufrufen von exponierten API-Methoden über das Frontend-HTTP und WebSocket die Endpunkte.

API Gateway-REST-API

Eine Sammlung von HTTP-Ressourcen und -Methoden, die in Backend-HTTP-Endpunkte, Lambda-Funktionen oder andere Dienste integriert sind. AWS Sie können diese Sammlung in einer oder mehreren Stufen bereitstellen. In der Regel werden API-Ressourcen in einer Ressourcenstruktur organisiert, die der Anwendungslogik entspricht. Jede API-Ressource kann

eine oder mehrere API-Methoden offenlegen, die eindeutige HTTP-Verben haben, die von API Gateway unterstützt werden. Weitere Informationen finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

API Gateway-HTTP-API

Eine Sammlung von Routen und Methoden, die mit Backend-HTTP-Endpunkten oder Lambda-Funktionen integriert sind. Sie können diese Sammlung in einer oder mehreren Stufen bereitstellen. Jede Route kann eine oder mehrere API-Methoden bereitstellen, die eindeutige HTTP-Verben haben, die von API Gateway unterstützt werden. Weitere Informationen finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

WebSocket API-Gateway-API

Eine Sammlung von WebSocket Routen und Routenschlüsseln, die in Backend-HTTP-Endpunkte, Lambda-Funktionen oder andere Dienste integriert sind. AWS Sie können diese Sammlung in einer oder mehreren Stufen bereitstellen. API-Methoden werden über WebSocket Frontend-Verbindungen aufgerufen, die Sie einem registrierten benutzerdefinierten Domainnamen zuordnen können.

API-Bereitstellung

Eine point-in-time Momentaufnahme Ihrer API-Gateway-API. Damit eine Bereitstellung von einem Client verwendet werden kann, muss sie mindestens einer API-Stufe zugeordnet sein.

API-Developer

Ihr AWS Konto, dem eine API Gateway Gateway-Bereitstellung gehört (z. B. ein Dienstanbieter, der auch programmatischen Zugriff unterstützt).

API-Endpunkt

Ein Hostname für eine API in API Gateway, die in einer bestimmten Region bereitgestellt wird. Der Hostname hat das Format `{api-id}.execute-api.{region}.amazonaws.com`. Die folgenden Arten von API-Endpunkten werden unterstützt:

- [Edge-optimierter API-Endpunkt](#)
- [Privater API-Endpunkt](#)
- [Regionaler API-Endpunkt](#)

API-Schlüssel

Eine alphanumerische Zeichenfolge, die API Gateway verwendet, um einen App-Entwickler zu identifizieren, der Ihre REST oder WebSocket API verwendet. API Gateway kann API-Schlüssel

für Sie erstellen. Alternativ können Sie Schlüssel aus einer CSV-Datei importieren. Sie können API-Schlüssel zusammen mit [Lambda-Autorisierern](#) oder [Nutzungsplänen](#) verwenden, um den Zugriff auf Ihre zu kontrollieren. APIs

Siehe [API-Endpunkte](#).

API-Eigentümer

Siehe [API-Developer](#).

API-Stufe

Ein logischer Verweis auf einen Lebenszyklusstatus Ihres API (z. B. "dev", "prod", "Beta", "v2"). API-Stufen werden mit API-ID und Stufenname bezeichnet.

App-Developer

Ein App-Ersteller, der möglicherweise über ein AWS Konto verfügt oder auch nicht und mit der API interagiert, die Sie als API-Entwickler bereitgestellt haben. App-Developer sind Ihre Kunden. Ein App-Developer wird in der Regel durch einen [API-Schlüssel](#) identifiziert.

Rückruf-URL

Wenn mit einem neuen Client über eine WebSocket Verbindung verbunden wird, können Sie eine Integration in API Gateway aufrufen, um die Callback-URL des Clients zu speichern. Sie können diese Rückruf-URL dann zum Senden von Nachrichten vom Backend-System zum Client verwenden.

Entwicklerportal

Eine Anwendung, mit der API-Anbieter ihre APIs und ihre API-Dokumentation an API-Nutzer weitergeben können. APIs sind in Produkte gruppiert, bei denen es sich um eine Sammlung von REST-API-Endpunkten, API-Dokumentation und ergänzender Produktdokumentation handelt.

Siehe [API Gateway Gateway-Portale](#).

Edge-optimierter API-Endpunkt

Der Standard-Hostname einer API-Gateway-API, die in der angegebenen Region bereitgestellt wird und dabei eine CloudFront Distribution verwendet, um den Client-Zugriff in der Regel aus verschiedenen AWS Regionen zu erleichtern. API-Anfragen werden an den nächstgelegenen CloudFront Point of Presence (POP) weitergeleitet, was in der Regel die Verbindungszeit für geografisch unterschiedliche Clients verbessert.

Siehe [API-Endpunkte](#).

Integrationsanforderung

Die interne Schnittstelle einer WebSocket API-Route oder REST-API-Methode in API Gateway, in der Sie den Hauptteil einer Routenanforderung oder die Parameter und den Hauptteil einer Methodenanforderung den vom Backend benötigten Formaten zuordnen.

Integrationsantwort

Die interne Schnittstelle einer WebSocket API-Route oder REST-API-Methode in API Gateway, in der Sie die vom Backend empfangenen Statuscodes, Header und Nutzdaten dem Antwortformat zuordnen, das an eine Client-App zurückgegeben wird.

Zuweisungsvorlage

Ein Skript in [Velocity Template Language \(VTL\)](#), mit dem der Anforderungstext im Frontend-Datenformat in das Backend-Datenformat konvertiert wird oder mit dem der Antworttext aus dem Backend-Datenformat in das Frontend-Datenformat konvertiert wird. Zuweisungsvorlagen können in der Integrationsanforderung oder in der Integrationsantwort angegeben werden. Sie können auf Daten verweisen, die zur Laufzeit als Kontext- und Stufenvariablen zur Verfügung gestellt werden.

Die Zuordnung kann so einfach wie eine [Identitätstransformation](#) sein, mit der für eine Anforderung Header oder Text unverändert durch die Integration vom Client an das Backend übergeben werden. Das Gleiche gilt für eine Antwort, in der die Nutzlast vom Backend an den Client übergeben wird.

Methodenanforderung

Die öffentliche Schnittstelle einer API-Methode in API Gateway, die die Parameter und den Text vorgibt, die ein App-Entwickler in Anfragen senden muss, um über die API auf das Backend zugreifen zu können.

Methodenantwort

Die öffentliche Schnittstelle einer REST-API, die Statuscodes, Header und Textmodelle vorgibt, die ein App-Developer in Antworten von der API erwarten sollte.

Pseudointegration

Bei einer Mock-Integration werden API-Antworten direkt von API Gateway generiert, ohne dass ein Integrations-Backend erforderlich ist. Als API-Entwickler entscheiden Sie, wie API Gateway auf eine Mock-Integrationsanfrage antwortet. Dazu konfigurieren Sie die Integrationsanforderung und -antwort der Methode, um einer Antwort einen bestimmten Statuscode zuzuweisen.

Model

Ein Datenschema, das die Datenstruktur einer Anforderungs- oder Antwortnutzlast angibt. Ein Modell ist erforderlich, um ein stark typisiertes SDK einer API zu generieren. Es wird auch zur Validierung der Nutzlast verwendet. Ein Modell bietet praktische Vorteile, da eine Beispiel-Zuweisungsvorlage generiert wird, über die eine Produktionszuweisungsvorlage initiiert werden kann. Für die Erstellung einer Zuweisungsvorlage ist ein Modell zwar hilfreich, aber nicht erforderlich.

Portal

[Siehe Entwicklerportal.](#)

Portal-Produkt

Ein Service oder eine Funktion, die Sie teilen möchten. Ihr Portalprodukt ist eine Sammlung von REST-Produktendpunkten und Produktseiten. Produkt-REST-Endpunkte sind die Zugriffspunkte zu Ihrem Portalprodukt. Sie bestehen aus dem Pfad und der Methode einer REST-API sowie der Phase, in der sie bereitgestellt wird. Bei Produktseiten handelt es sich um eine Dokumentation, die Sie bereitstellen, um zu erklären, wie API-Nutzer Ihre Produktendpunkte verwenden können. Sie können Produkte AWS kontenübergreifend teilen, um sie zu Portalen hinzuzufügen.

Weitere Informationen finden [Sie unter Produkt erstellen.](#)

Private API

Siehe [Privater API-Endpunkt.](#)

Privater API-Endpunkt

Ein API-Endpunkt, der über Schnittstellen-VPC-Endpunkte bereitgestellt wird und einem Client den sicheren Zugriff auf private API-Ressourcen innerhalb einer VPC ermöglicht. Private APIs sind vom öffentlichen Internet isoliert und können nur über VPC-Endpunkte für API Gateway abgerufen werden, denen Zugriff gewährt wurde.

Private Integration

Ein API Gateway-Integrationstyp für einen Client zum Zugriff auf Ressourcen in der VPC eines Kunden über einen privaten REST-API-Endpunkt, ohne die Ressourcen im öffentlichen Internet offenzulegen.

Proxy-Integration

Eine vereinfachte API Gateway-Integrationskonfiguration. Sie können eine Proxy-Integration als HTTP-Proxy-Integration oder als Lambda-Proxy-Integration einrichten.

Bei einer HTTP-Proxy-Integration leitet API Gateway die gesamte Anfrage und Antwort zwischen dem Frontend und einem HTTP-Backend weiter. Bei der Lambda-Proxy-Integration sendet API Gateway die gesamte Anfrage als Eingabe an eine Lambda-Funktion im Backend. API Gateway wandelt dann die Ausgabe der Lambda-Funktion in eine HTTP-Antwort des Frontends um.

In REST APIs wird die Proxyintegration am häufigsten mit einer Proxyressource verwendet, die durch eine Greedy Path-Variable (z. B. {proxy+}) in Kombination mit einer ANY Catch-All-Methode dargestellt wird.

Quick Create

Sie können die Schnellerstellung verwenden, um die Erstellung einer HTTP-API zu vereinfachen. Quick Create erstellt eine API mit einer Lambda- oder HTTP-Integration, eine Catch-All-Standardroute und eine Standardphase, die für die automatische Bereitstellung von Änderungen konfiguriert ist. Weitere Informationen finden Sie unter [the section called “Erstellen Sie eine HTTP-API mithilfe der AWS CLI”](#).

Regionaler API-Endpunkt

Der Hostname einer API, die in der angegebenen Region bereitgestellt wird und für Clients, z. B. EC2 Instances, in derselben Region vorgesehen ist. AWS API-Anfragen werden direkt an die regionsspezifische API Gateway gerichtet, ohne dass eine Verteilung erforderlich ist. CloudFront Bei Anfragen innerhalb einer Region umgeht ein regionaler Endpunkt den unnötigen Roundtrip zu einer Distribution. CloudFront

Darüber hinaus können Sie [latenzbasiertes Routing](#) an den regionalen Endpunkten anwenden, um eine API für mehrere Regionen mithilfe derselben regionalen API-Endpunkt-Konfiguration bereitzustellen, dieselben benutzerdefinierten Domännennamen für jede bereitgestellte API festzulegen und latenzbasierte DNS-Datensätze in Route 53 zu konfigurieren, um Client-Anfragen an die Region mit der niedrigsten Latenz zu leiten.

Siehe [API-Endpunkte](#).

Route

Eine WebSocket Route in API Gateway wird verwendet, um eingehende Nachrichten basierend auf dem Inhalt der Nachricht an eine bestimmte Integration, z. B. eine AWS Lambda Funktion, weiterzuleiten. Wenn Sie Ihre WebSocket API definieren, geben Sie einen Routenschlüssel und ein Integrations-Backend an. Der Routenschlüssel ist ein Attribut im Nachrichtentext. Wenn der Routenschlüssel in einer eingehenden Nachricht abgeglichen wird, wird das Integrations-Backend aufgerufen.

Es kann auch eine Standardroute für nicht übereinstimmende Routenschlüssel oder zur Angabe eines Proxy-Modells festgelegt werden, mit dem die Nachricht unverändert an Backend-Komponenten übergeben wird, die die Weiterleitung und Verarbeitung der Anforderung durchführen.

Routenanforderung

Die öffentliche Schnittstelle einer WebSocket API-Methode in API Gateway, die den Text definiert, den ein App-Entwickler in den Anfragen senden muss, um über die API auf das Backend zuzugreifen.

Routenantwort

Die öffentliche Schnittstelle einer WebSocket API, die die Statuscodes, Header und Body-Modelle definiert, die ein App-Entwickler von API Gateway erwarten sollte.

Nutzungsplan

Ein [Nutzungsplan](#) bietet ausgewählten API-Clients Zugriff auf einen oder mehrere bereitgestellte REST- oder WebSocket APIs. Sie können einen Nutzungsplan dazu verwenden, die Drosselungs- und Kontingentlimits zu konfigurieren, die für einzelne Client-API-Schlüssel erzwungen werden.

WebSocket Verbindung

API Gateway unterhält eine beständige Verbindung zwischen Clients und API Gateway. Es gibt keine beständige Verbindung zwischen API Gateway und Backend-Integrationen wie z. B. Lambda-Funktionen. Backend-Services werden nach Bedarf aufgerufen, basierend auf dem Inhalt der Nachrichten, die von Clients empfangen werden.

Wählen Sie zwischen REST APIs und HTTP APIs

REST APIs und HTTP APIs sind beide RESTful API-Produkte. REST APIs unterstützt mehr Funktionen als HTTP APIs, während HTTP APIs mit minimalen Funktionen konzipiert wurde, sodass sie zu einem niedrigeren Preis angeboten werden können. Wählen Sie REST, APIs wenn Sie Funktionen wie API-Schlüssel, Drosselung pro Client, Validierung von Anfragen, AWS WAF Integration oder private API-Endpunkte benötigen. Wählen Sie HTTP APIs, wenn Sie die in REST enthaltenen Funktionen nicht benötigen. APIs

In den folgenden Abschnitten werden die wichtigsten Funktionen zusammengefasst, die in REST APIs und HTTP APIs verfügbar sind. Der Einfachheit halber werden zusätzliche Links bereitgestellt,

um zwischen den Abschnitten zu REST-API und HTTP-API im API Gateway-Entwicklerhandbuch zu navigieren.

Endpunkttyp

Der Endpunkttyp bezieht sich auf den Endpunkt, den API Gateway für Ihre API erstellt. Weitere Informationen finden Sie unter [the section called “API-Gateway-Endpunkttypen”](#).

Endpunkttypen	REST-API	HTTP-API
Edge-optimiert	Ja	Nein
Regional	Ja	Ja
Privat	Ja	Nein

Sicherheit

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, z. B. vor böswilligen Akteuren oder Spitzeln im Datenverkehr. Weitere Informationen hierzu finden Sie unter [the section called “Schutz”](#) und [the section called “Schutz”](#).

Sicherheitsfunktionen	REST-API	HTTP-API
Gegenseitige TLS-Authentifizierung	Ja	Ja
Zertifikate für Backend-Authentifizierung	Ja	Nein
AWS WAF	Ja	Nein

Autorisierung

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre API. Weitere Informationen erhalten Sie unter [the section called “Zugriffskontrolle”](#) und [the section called “Zugriffskontrolle”](#).

Autorisierungsoptionen	REST-API	HTTP-API
IAM	Ja	Ja
Ressourcenrichtlinien	Ja	Nein
Amazon Cognito	Ja	Ja ¹
Benutzerdefinierte Autorisierung mit einer Funktion AWS Lambda	Ja	Ja
JSON Web Token (JWT) ²	Nein	Ja

¹ Sie können Amazon Cognito mit einem [JWT-Genehmiger](#) verwenden.

² Sie können einen [Lambda-Autorisierer verwenden, um JWTs für REST](#) zu validieren. APIs

API-Management

Wählen Sie REST APIs , wenn Sie API-Verwaltungsfunktionen wie API-Schlüssel und Ratenbegrenzung pro Client benötigen. Weitere Informationen finden Sie unter [the section called “Verteilen”](#), [the section called “Benutzerdefinierte Domänennamen”](#) und [the section called “Benutzerdefinierte Domainnamen”](#).

Features	REST-API	HTTP-API
Benutzerdefinierte Domänen	Ja	Ja
API-Schlüssel	Ja	Nein
Clientbasierte Ratenbegrenzung	Ja	Nein
Clientbasierte Nutzungsdrosselung	Ja	Nein
Entwicklerportal	Ja	Nein

Entwicklung

Während Sie Ihre API Gateway-API entwickeln, entscheiden Sie sich für eine Reihe von Merkmalen Ihrer API. Diese Eigenschaften hängen davon ab, wofür Ihre API verwendet werden soll. Weitere Informationen finden Sie unter [the section called “Entwickeln”](#) und [the section called “Entwickeln”](#).

Features	REST-API	HTTP-API
CORS-Konfiguration	Ja	Ja
Testaufrufe	Ja	Nein
Caching	Ja	Nein
Benutzergesteuerte Bereitstellungen	Ja	Ja
Automatische Bereitstellungen	Nein	Ja
Benutzerdefinierte Gateway-Antworten	Ja	Nein
Canary-Release-Bereitstellungen	Ja	Nein
Anforderungvalidierung	Ja	Nein
Transformation von Anfrageparametern	Ja	Ja
Transformation von Anforderungstext	Ja	Nein

Überwachen

API Gateway unterstützt verschiedene Optionen zum Protokollieren von API-Anfragen und zur Überwachung Ihrer APIs. Weitere Informationen erhalten Sie unter [the section called “Überwachen”](#) und [the section called “Überwachen”](#).

Feature	REST-API	HTTP-API
CloudWatch Amazon-Metriken	Ja	Ja
Zugriff auf Logs auf CloudWatch Logs	Ja	Ja
Zugriffsprotokolle für Amazon Data Firehose	Ja	Nein
Ausführungsprotokolle	Ja	Nein
AWS X-Ray Rückverfolgung	Ja	Nein

Integrationen

Integrationen verbinden Ihre API-Gateway-API mit Backend-Ressourcen. Weitere Informationen erhalten Sie unter [the section called “Integrationen”](#) und [the section called “Integrationen”](#).

Feature	REST-API	HTTP-API
Öffentliche HTTP-Endpunkte	Ja	Ja
AWS Dienstleistungen	Ja	Ja
AWS Lambda Funktionen	Ja	Ja
Private Integrationen mit Network Load Balancers	Ja	Ja
Private Integrationen mit Application Load Balancers	Ja	Ja
Private Integrationen mit AWS Cloud Map	Nein	Ja
Pseudointegrationen	Ja	Nein

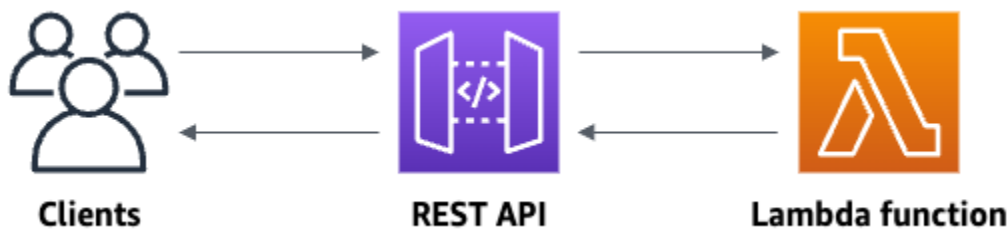
Feature	REST-API	HTTP-API
Antwort-Streaming	Ja	Nein

Erste Schritte mit der REST-API-Konsole

In dieser Übung zu den ersten Schritten erstellen Sie eine Serverless-REST-API mithilfe der API-Gateway-REST-API-Konsole. Mit Serverless APIs können Sie sich auf Ihre Anwendungen konzentrieren, anstatt Ihre Zeit mit der Bereitstellung und Verwaltung von Servern zu verbringen. Diese Übung dauert weniger als 20 Minuten und kann im Rahmen des [Kostenlosen AWS - Kontingents](#) durchgeführt werden.

Zunächst erstellen Sie eine Lambda-Funktion unter Verwendung der Lambda-Konsole. Dann erstellen Sie eine REST-API mit der API-Gateway-REST-API-Konsole. Anschließend erstellen Sie eine API-Methode und integrieren sie mithilfe einer Lambda-Proxy-Integration in eine Lambda-Funktion. Schließlich stellen Sie Ihre API bereit und rufen sie auf.

Wenn Sie Ihre REST-API aufrufen, leitet API Gateway die Anfrage an Ihre Lambda-Funktion weiter. Lambda führt die Funktion aus und gibt eine Antwort an API Gateway zurück. API Gateway gibt dann diese Antwort an Sie zurück.



Um diese Übung durchführen zu können, benötigen Sie einen AWS-Konto und einen AWS Identity and Access Management (IAM) -Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Einrichten der API Gateway](#).

Themen

- [Schritt 1: Erstellen einer Lambda-Funktion](#)
- [Schritt 2: Erstellen einer REST-API](#)
- [Schritt 3: Erstellen einer Lambda-Proxy-Integration](#)
- [Schritt 4: Bereitstellen Ihrer API](#)

- [Schritt 5: Aufrufen Ihrer API](#)
- [\(Optional\) Schritt 6: Bereinigen](#)

Schritt 1: Erstellen einer Lambda-Funktion

Sie verwenden eine Lambda-Funktion für das Backend Ihrer API. Lambda führt Ihren Code nur bei Bedarf aus und skaliert automatisch – von einigen Anforderungen pro Tag bis zu Tausenden pro Sekunde.

In dieser Übung verwenden Sie eine Node.js-Standardfunktion in der Lambda-Konsole.

So erstellen Sie eine Lambda-Funktion

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie Funktion erstellen.
3. Geben Sie unter Basic Information (Grundlegende Informationen) für Function name (Funktionsname) **my-function** ein.
4. Verwenden Sie die Standardeinstellung für alle anderen Optionen.
5. Wählen Sie Funktion erstellen.

Der standardmäßige Lambda-Funktionscode sollte wie folgt aussehen:

```
export const handler = async (event) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify('The API Gateway REST API console is great!'),
  };
  return response;
};
```

Sie können Ihre Lambda-Funktion für diese Übung ändern, solange die Antwort der Funktion mit dem [Format übereinstimmt, das API Gateway benötigt](#).

Ersetzen Sie den Standardantworttext (Hello from Lambda!) durch The API Gateway REST API console is great!. Wenn Sie die Beispielfunktion aufrufen, gibt sie zusammen mit der aktualisierten Antwort eine 200-Antwort an die Clients zurück.

Schritt 2: Erstellen einer REST-API

Als Nächstes erstellen Sie eine REST-API mit einer Root-Ressource (/).

So erstellen Sie eine REST-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Führen Sie eine der folgenden Aktionen aus:
 - Um Ihre erste API zu erstellen, wählen Sie für REST-API die Option Erstellen.
 - Wenn Sie zuvor eine API erstellt haben, wählen Sie API erstellen und dann für REST-API die Option Erstellen.
3. Geben Sie in API name (API-Name) **my-rest-api** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Schritt 3: Erstellen einer Lambda-Proxy-Integration

Als Nächstes erstellen Sie eine API-Methode für Ihre REST-API in der Root-Ressource (/) und integrieren die Methode mithilfe einer Proxy-Integration in Ihre Lambda-Funktion. Bei einer Lambda-Proxy-Integration leitet API Gateway die eingehende Anfrage vom Client direkt an die Lambda-Funktion weiter.

So erstellen Sie eine Lambda-Proxy-Integration

1. Wählen Sie die /-Ressource und dann Methode erstellen aus.
2. Wählen Sie als Methodentyp die Option ANY aus.
3. Wählen Sie für den Integrationstyp die Option Lambda aus.
4. Schalten Sie die Lambda-Proxy-Integration ein.
5. Geben Sie **my-function** für Lambda-Funktion ein und wählen Sie dann Ihre Lambda.Funktion aus.
6. Wählen Sie Methode erstellen aus.

Schritt 4: Bereitstellen Ihrer API

Als Nächstes erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Stufe.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **Prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

Jetzt können Clients Ihre API aufrufen. Um Ihre API vor der Bereitstellung zu testen, können Sie optional die ANY-Methode wählen, zur Registerkarte Test navigieren und dann Test auswählen.

Schritt 5: Aufrufen Ihrer API

So rufen Sie Ihre API auf

1. Wählen Sie im Haupt-Navigationsbereich Stufe aus.
2. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Stage details [Info](#)

Stage name
Prod

Cache cluster [Info](#)
⊖ Inactive

Default method-level caching
⊖ Inactive

Rate [Info](#)
10000

Burst [Info](#)
5000

Web ACL
-

Client certificate
-

Invoke URL
 `https://abcd1234.execute-api.us-east-1.amazonaws.com/Prod`

Active deployment
efg456 on June 01, 2023, 15:00 (UTC-07:00)

[Edit](#)

3. Geben Sie die Aufruf-URL in einen Webbrowser ein.

Die URL sollte wie folgt aussehen: `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Ihr Browser sendet eine GET-Anforderung an die API.

4. Verifizieren Sie die Antwort Ihrer API. Sie sollten den Text "The API Gateway REST API console is great!" in Ihrem Browser sehen.

(Optional) Schritt 6: Bereinigen

Um zu verhindern, dass Ihnen unnötige Kosten entstehen AWS-Konto, löschen Sie die Ressourcen, die Sie im Rahmen dieser Übung erstellt haben. In den folgenden Schritten werden Ihre REST-API, Ihre Lambda-Funktion und die zugehörigen Ressourcen gelöscht.

So löschen Sie Ihre REST-API

1. Wählen Sie im Bereich Ressourcen die Option API-Aktionen und API löschen aus.
2. Geben Sie im Dialogfeld API löschen Bestätigen ein und wählen Sie dann Löschen.

So löschen Sie Ihre Lambda-Funktion

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie auf der Seite Funktionen Ihre Funktion aus. Wählen Sie Aktionen, Löschen aus.
3. Geben Sie im Dialogfeld Löschen den Text **delete** ein und wählen Sie dann Löschen aus.

So löschen Sie die Protokollgruppe Ihrer Lambda-Funktion

1. Öffnen Sie die [Seite Protokollgruppen](#) der CloudWatch Amazon-Konsole.
2. Wählen Sie auf der Seite Protokollgruppen die Protokollgruppe (/aws/lambda/my-function) Ihrer Funktion aus. Wählen Sie für Aktionen die Option Protokollgruppe(n) löschen aus.
3. Wählen Sie im Dialogfeld Delete log group(s) (Protokollgruppe(n) löschen) die Option Delete (Löschen) aus.

So löschen Sie die Ausführungsrolle Ihrer Lambda-Funktion:

1. Öffnen Sie die Seite [Roles \(Rollen\)](#) in der IAM-Konsole.

2. (Optional) Geben Sie auf der Seite Rollen in das Suchfeld **my-function** ein.
3. Wählen Sie die Rolle Ihrer Funktion aus (z. B. my-function-*31exxmpl*) und wählen Sie dann Löschen aus.
4. Geben Sie im Dialogfeld **my-function-31exxmpl** löschen? den Namen der Rolle ein und wählen Sie anschließend Löschen aus.

 Tip

Sie können die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie AWS CloudFormation oder AWS Serverless Application Model (AWS SAM) verwenden. Einige CloudFormation Beispielvorlagen finden Sie in den [Beispielvorlagen für API Gateway](#) im awsdocs-Repository GitHub .

Einrichten der API Gateway

Bevor Sie Amazon API Gateway zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus.

Melde dich an für ein AWS-Konto

Um loszulegen AWS, benötigen Sie eine AWS-Konto. Informationen zum Erstellen eines AWS-Konto finden Sie unter [Erste Schritte mit einem AWS-Konto](#) im AWS -Kontenverwaltung Referenzhandbuch.

Erste Schritte mit API-Gateway

Bei dieser Übung „Erste Schritte“ erstellen Sie eine Serverless-API. Mit Serverless-APIs können Sie sich auf Ihre Anwendungen konzentrieren, anstatt Zeit mit der Bereitstellung und Verwaltung von Servern zu verbringen. Diese Übung dauert weniger als 20 Minuten und kann im Rahmen des [AWSkostenlosen Kontingents](#) durchgeführt werden.

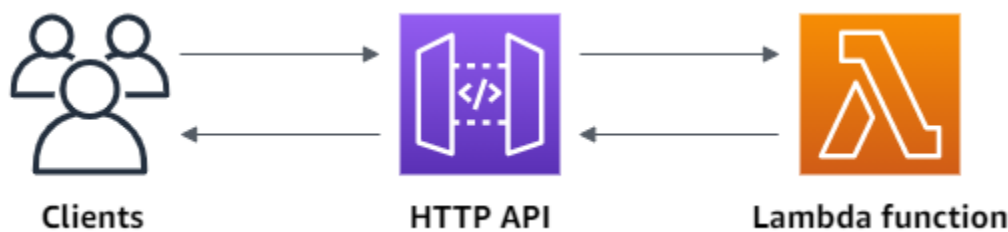
Zuerst erstellen Sie mithilfe der AWS Lambda-Konsole eine Lambda-Funktion. Als Nächstes erstellen Sie eine HTTP-API mit der API-Gateway-Konsole. Dann rufen Sie Ihre API auf.

Note

In dieser Übung wird eine HTTP-API verwendet. API-Gateway unterstützt auch REST-APIs, die mehr Funktionen enthalten. Ein Tutorial zur Verwendung einer REST-API finden Sie unter [the section called “Erste Schritte mit der REST-API-Konsole”](#).

Weitere Informationen zu den Unterschieden zwischen HTTP-APIs und REST-APIs finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

Wenn Sie Ihre HTTP-API aufrufen, leitet API Gateway die Anfrage an Ihre Lambda-Funktion weiter. Lambda führt die Lambda-Funktion aus und gibt eine Antwort an API Gateway zurück. API Gateway gibt dann eine Antwort an Sie zurück.



Um diese Übung abzuschließen, benötigen Sie ein AWS-Konto und einen AWS Identity and Access Management-Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

Themen

- [Schritt 1: Erstellen einer Lambda-Funktion](#)
- [Schritt 2: HTTP-API erstellen](#)
- [Schritt 3: Ihre API testen](#)
- [\(Optional\) Schritt 4: Bereinigen](#)

- [Nächste Schritte](#)

Schritt 1: Erstellen einer Lambda-Funktion

Sie verwenden eine Lambda-Funktion für das Backend Ihrer API. Lambda führt Ihren Code nur bei Bedarf aus und skaliert automatisch – von einigen Anforderungen pro Tag bis zu Tausenden pro Sekunde.

In diesem Beispiel verwenden Sie die Standardfunktion Node.js von der Lambda-Konsole aus.

So erstellen Sie eine Lambda-Funktion:

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie Create function (Funktion erstellen).
3. Geben Sie für Function name (Funktionsname) **my-function** ein.
4. Verwenden Sie die Standardeinstellung für alle anderen Optionen.
5. Wählen Sie Create function (Funktion erstellen).

Die Beispielfunktion gibt eine 200-Antwort an Clients und den Text Hello from Lambda! zurück.

Sie können Ihre Lambda-Funktion ändern, solange die Antwort der Funktion mit dem [Format übereinstimmt, das API Gateway benötigt](#).

Der standardmäßige Lambda-Funktionscode sollte wie folgt aussehen:

```
export const handler = async (event) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify('Hello from Lambda!'),
  };
  return response;
};
```

Schritt 2: HTTP-API erstellen

Als Nächstes erstellen Sie eine HTTP-API. API Gateway unterstützt auch REST-APIs und WebSocket-APIs, aber eine HTTP-API ist die beste Wahl für diese Übung. REST-APIs unterstützen

mehr Funktionen als HTTP-APIs, aber wir benötigen diese Funktionen für diese Übung nicht. HTTP-APIs sind mit minimalen Funktionen ausgestattet, so dass sie zu einem niedrigeren Preis angeboten werden können. WebSocket-APIs unterhalten dauerhafte Verbindungen mit Clients für die Vollduplex-Kommunikation, was für dieses Beispiel nicht erforderlich ist.

Die HTTP-API bietet einen HTTP-Endpunkt für Ihre Lambda-Funktion. API Gateway leitet Anfragen an Ihre Lambda-Funktion weiter und gibt dann die Antwort der Funktion an Clients zurück.

So erstellen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Führen Sie eine der folgenden Aufgaben aus:
 - Um Ihre erste API für HTTP-API zu erstellen, wählen Sie Build (Erstellen).
 - Wenn Sie zuvor eine API erstellt haben, wählen Sie Create API (API erstellen) und dann für HTTP API Build (Erstellen).
3. Wählen Sie für Integrations (Integrationen) die Option Add integration (Integration hinzufügen) aus.
4. Wählen Sie Lambda.
5. Geben Sie für die Lambda function (Lambda-Funktion) **my-function** ein.
6. Geben Sie unter API name (API-Name) **my-http-api** ein.
7. Wählen Sie Next.
8. Überprüfen Sie die Route, die API Gateway für Sie erstellt, und wählen Sie dann Next (Weiter).
9. Überprüfen Sie die Phase, die API Gateway für Sie erstellt, und wählen Sie dann Next (Weiter).
10. Wählen Sie Create aus.

Jetzt haben Sie eine HTTP-API mit einer Lambda-Integration erstellt, die bereit ist, Anfragen von Clients zu erhalten.

Schritt 3: Ihre API testen

Als Nächstes testen Sie Ihre API, um sicherzustellen, dass sie funktioniert. Verwenden Sie zur Vereinfachung einen Webbrowser, um Ihre API aufzurufen.

So testen Sie Ihre API

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie Ihre API aus.
3. Notieren Sie sich die Aufruf-URL Ihrer API.

my-http-api

Stage: - ▼

Deploy

API details

Edit

API ID abcd1234	Protocol HTTP	Created 2023-10-30
Description My first HTTP API	Default endpoint ✔ Enabled https://abcd1234.execute-api.us-west-1.amazonaws.com	ARN arn:aws:apigateway:us-west-1::apis/abcd1234

Stages for my-http-api (1)

Find resources

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://abcd1234.execute-api.us-west-1.amazonaws.com	aaa111	enabled	2023-10-30

4. Kopieren Sie die Aufruf-URL Ihrer API und geben Sie sie in einen Webbrowser ein. Hängen Sie den Namen Ihrer Lambda-Funktion an Ihre Aufruf-URL an, um Ihre Lambda-Funktion aufzurufen. Standardmäßig erstellt die API-Gateway-Konsole eine Route mit dem gleichen Namen wie Ihre Lambda-Funktion, my-function.

Die URL sollte wie folgt aussehen: `https://abcdef123.execute-api.us-east-2.amazonaws.com/my-function`.

Ihr Browser sendet eine GET-Anforderung an die API.

5. Verifizieren Sie die Antwort Ihrer API. Sie sollten den Text "Hello from Lambda!" in Ihrem Browser sehen.

(Optional) Schritt 4: Bereinigen

Um unnötige Kosten zu vermeiden, löschen Sie die Ressourcen, die Sie im Rahmen dieser Übung „Erste Schritte“ erstellt haben. In den folgenden Schritten werden Ihre HTTP-API, Ihre Lambda-Funktion und die zugehörigen Ressourcen gelöscht.

So löschen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie auf der Seite APIs eine API aus. Wählen Sie Actions und anschließend Delete.
3. Wählen Sie Delete (Löschen).

So löschen Sie eine Lambda-Funktion

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie auf der Seite Functions (Funktionen) eine Funktion aus. Wählen Sie Actions und anschließend Delete.
3. Wählen Sie Delete (Löschen).

So löschen Sie die Protokollgruppe einer Lambda-Funktion

1. Öffnen Sie in der Amazon CloudWatch-Konsole die Seite [Log groups \(Protokollgruppen\)](#).
2. Wählen Sie auf der Seite Log groups (Protokollgruppen) die Protokollgruppe (/aws/lambda/my-function) der Funktion aus. Wählen Sie Actions (Aktionen) und dann Delete log group (Protokollgruppe löschen) aus.
3. Wählen Sie Delete (Löschen).

So löschen Sie die Ausführungsrolle einer Lambda-Funktion

1. Öffnen Sie in der AWS Identity and Access Management-Konsole die Seite [Rollen](#).
2. Wählen Sie die Rolle der Funktion aus, zum Beispi, my-function-*31exxmpl*.
3. Wählen Sie Delete role (Rolle löschen) aus.
4. Wählen Sie Yes, delete (Ja, löschen) aus.

Sie können die Erstellung und Bereinigung von AWS-Ressourcen mithilfe von CloudFormation oder AWS SAM automatisieren. Unter [CloudFormation-Beispielvorlagen](#) finden Sie Beispiele für CloudFormation-Vorlagen.

Nächste Schritte

In diesem Beispiel haben Sie die AWS-Managementkonsole verwendet, um eine einfache HTTP-API zu erstellen. Die HTTP-API ruft eine Lambda-Funktion auf und gibt eine Antwort an Clients zurück.

Im Folgenden sind die nächsten Schritte aufgeführt, wenn Sie weiterhin mit API Gateway arbeiten.

- [Konfigurieren Sie zusätzliche Arten von API-Integrationen](#), einschließlich:
 - [HTTP-Endpunkte](#)
 - [Private Ressourcen in einer VPC, wie Amazon ECS Services](#)
 - [AWS-Services wie Amazon Simple Queue Service, AWS Step Functions und Kinesis Data Streams](#)
- [Kontrollieren des Zugriffs auf Ihre APIs](#)
- [Aktivieren der Protokollierung für Ihre Instance](#)
- [Konfigurieren der Drosselung für Ihre APIs](#)
- [Konfigurieren benutzerdefinierter Domänen für Ihre APIs](#)

Hilfe zu Amazon API Gateway aus der Community erhalten Sie im [API Gateway-Diskussionsforum](#). Wenn Sie dieses Forum besuchen, verlangt AWS möglicherweise, dass Sie sich anmelden.

Hilfe zu API Gateway direkt von AWS erhalten Sie über die Support-Optionen auf der [AWS-Support-Seite](#).

Weitere Informationen finden Sie außerdem in unseren [häufig gestellten Fragen](#) (FAQs). Außerdem können Sie uns [direkt kontaktieren](#).

Amazon API Gateway-Tutorials und -Workshops

Die folgenden Tutorials und Workshops bieten praktische Übungen, die Ihnen beim Kennenlernen von API Gateway helfen. Wir empfehlen Ihnen, zunächst [the section called “Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration”](#) abzuschließen.

REST API-Tutorials

- [Wählen Sie ein AWS Lambda Integrations-Tutorial](#)
- [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#)
- [Wählen Sie ein Tutorial zur HTTP-Integration aus](#)
- [Tutorial: Erstellen einer REST-API mit einer privaten Integration](#)
- [Tutorial: Erstellen Sie eine REST-API mit einer AWS Integration](#)
- [Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration](#)
- [Tutorial: REST-API als AmazonS3-Proxy erstellen](#)
- [Tutorial: REST-API als Amazon Kinesis-Proxy erstellen](#)
- [Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI](#)
- [Tutorial: Erstellen einer privaten REST-API](#)

HTTP API-Tutorials

- [Tutorial: Erstellen einer CRUD-HTTP-API mit Lambda und DynamoDB](#)
- [Tutorial: Erstellen einer HTTP-API mit privater Integration für einen Amazon-ECS-Service](#)

WebSocket-API-Tutorials

- [Tutorial: Erstellen Sie eine WebSocket Chat-App mit einer WebSocket API, Lambda und DynamoDB](#)

Workshops

- [Erstellen einer Serverless-Webanwendung](#)
- [CI/CD für Serverless-Anwendungen](#)

- [Workshop für Serverless-Sicherheit](#)
- [Serverless-Identitätsverwaltung, -Authentifizierung und -Autorisierung](#)
- [Der Amazon-API-Gateway-Workshop](#)

Amazon API Gateway-REST-API-Tutorials

Die folgenden Tutorials enthalten praktische Übungen, die Ihnen helfen, mehr über API Gateway REST APIs zu erfahren.

Themen

- [Wählen Sie ein AWS Lambda Integrations-Tutorial](#)
- [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#)
- [Wählen Sie ein Tutorial zur HTTP-Integration aus](#)
- [Tutorial: Erstellen einer REST-API mit einer privaten Integration](#)
- [Tutorial: Erstellen Sie eine REST-API mit einer AWS Integration](#)
- [Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration](#)
- [Tutorial: REST-API als AmazonS3-Proxy erstellen](#)
- [Tutorial: REST-API als Amazon Kinesis-Proxy erstellen](#)
- [Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI](#)
- [Tutorial: Erstellen einer privaten REST-API](#)

Wählen Sie ein AWS Lambda Integrations-Tutorial

Um eine API mit Lambda-Integrationen zu erstellen, können Sie die Lambda-Proxy-Integration oder die Lambda-Non-Proxy-Integration verwenden.

Bei der Lambda-Proxy-Integration kann die Eingabe für die Lambda-Funktion als eine beliebige Kombination von Anforderungs-Headern, Pfadvariablen, Abfragezeichenfolgen-Parametern, Text und API-Konfigurationsdaten ausgedrückt werden. Sie wählen ganz einfach eine Lambda-Funktion aus. API Gateway konfiguriert die Integrationsanforderung und Integrationsantwort für Sie. Sobald sie eingerichtet ist, kann sich Ihre API-Methode ohne Änderungen an den vorhandenen Einstellungen weiterentwickeln. Dies ist möglich, da die Backend-Lambda-Funktion die eingehenden Anforderungsdaten analysiert und dem Client antwortet.

Bei der Lambda-Nicht-Proxy-Integration müssen Sie sicherstellen, dass die Eingabe an die Lambda-Funktion als Payload der Integrationsanforderung geliefert wird. Sie müssen alle vom Client bereitgestellten Eingabedaten zum entsprechenden Integrationsanforderungstext als Anforderungsparameter zuordnen. Möglicherweise müssen Sie auch den vom Client bereitgestellten Anfragetextkörper in ein Format umwandeln, das von der Lambda-Funktion erkannt wird.

In einem Lambda-Proxy- oder einer Lambda-Nicht-Proxy-Integration können Sie eine Lambda-Funktion in einem anderen Konto als dem Konto verwenden, in dem Sie Ihre API erstellt haben.

Themen

- [Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration](#)
- [Tutorial: Erstellen einer REST-API mit einer Lambda-Nicht-Proxy-Integration](#)
- [Tutorial: REST-API mit kontenübergreifender Lambda-Proxy-Integration erstellen](#)

Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration

[Lambda-Proxy-Integration](#) ist ein einfacher, flexibler API Gateway API-Integrationstyp, der es Ihnen ermöglicht, eine API-Methode – oder eine ganze API – mit einer Lambda-Funktion zu integrieren. Die Lambda-Funktion kann in [jeder Sprache, die Lambda unterstützt](#) geschrieben werden. Da es sich um eine Proxy-Integration handelt, können Sie die Implementierung der Lambda-Funktion jederzeit ändern, ohne Ihre API erneut bereitstellen zu müssen.

In diesem Tutorial führen Sie folgende Aufgaben aus:

- Erstellen Sie eine "Hello, World!"- Lambda-Funktion ist das Backend für die API.
- Erstellen und testen Sie eine "Hello, World!"- API mit Lambda-Proxy-Integration.

Themen

- [Erstellen Sie eine "Hello, World!"- Lambda-Funktion](#)
- [Erstellen Sie eine "Hello, World!"- API](#)
- [Bereitstellen und Testen der API](#)

Erstellen Sie eine "Hello, World!"- Lambda-Funktion

Erstellen von „Hello, World!“ Lambda-Funktion in der Lambda-Konsole

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.

2. Wählen Sie in der AWS Navigationsleiste eine [AWS-Region](#) aus.

Note

Notieren Sie die Region, in der Sie die Lambda-Funktion erstellen. Sie benötigen sie zum Erstellen der API.

3. Wählen Sie im Navigationsbereich Functions (Funktionen) aus.
4. Wählen Sie Create function (Funktion erstellen).
5. Wählen Sie Author from scratch aus.
6. Führen Sie unter Basic information (Grundlegende Informationen) die folgenden Schritte aus:
 - a. Geben Sie in Function name (Funktionsname) **GetStartedLambdaProxyIntegration** ein.
 - b. Wählen Sie für Laufzeit die neueste unterstützte Node.js- oder Python-Laufzeit aus.
 - c. Behalten Sie für Architektur die Standardeinstellung bei.
 - d. Erweitern Sie unter Berechtigungen die Option Standardausführungsrolle ändern. Wählen Sie in der Dropdown-Liste Ausführungsrolle die Option Neue Rolle aus AWS - Richtlinienvorlagen erstellen aus.
 - e. Geben Sie in Role Name (Rollenname) den Namen **GetStartedLambdaBasicExecutionRole** ein.
 - f. Lassen Sie das Feld Policy templates (Richtlinien-Vorlagen) leer.
 - g. Wählen Sie Create function (Funktion erstellen).
7. Im Inline-Code-Editor finden Sie unter Funktionscode copy/paste den folgenden Code:

Node.js

```
export const handler = async(event, context) => {
  console.log('Received event:', JSON.stringify(event, null, 2));
  var res ={
    "statusCode": 200,
    "headers": {
      "Content-Type": "*/*"
    }
  };
  var greeter = 'World';
  if (event.greeter && event.greeter!="") {
```

```

    greeter = event.greeter;
  } else if (event.body && event.body !== "") {
    var body = JSON.parse(event.body);
    if (body.greeter && body.greeter !== "") {
      greeter = body.greeter;
    }
  } else if (event.queryStringParameters &&
event.queryStringParameters.greeter && event.queryStringParameters.greeter !==
"") {
    greeter = event.queryStringParameters.greeter;
  } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
event.multiValueHeaders.greeter !== "") {
    greeter = event.multiValueHeaders.greeter.join(" and ");
  } else if (event.headers && event.headers.greeter && event.headers.greeter !
= "") {
    greeter = event.headers.greeter;
  }
  res.body = "Hello, " + greeter + "!";
  return res
};

```

Python

```

import json

def lambda_handler(event, context):
    print(event)

    greeter = 'World'

    try:
        if (event['queryStringParameters']) and (event['queryStringParameters']
['greeter']) and (
            event['queryStringParameters']['greeter'] is not None):
            greeter = event['queryStringParameters']['greeter']
    except KeyError:
        print('No greeter')

    try:
        if (event['multiValueHeaders']) and (event['multiValueHeaders']
['greeter']) and (
            event['multiValueHeaders']['greeter'] is not None):

```

```
        greeter = " and ".join(event['multiValueHeaders']['greeter'])
except KeyError:
    print('No greeter')

try:
    if (event['headers']) and (event['headers']['greeter']) and (
        event['headers']['greeter'] is not None):
        greeter = event['headers']['greeter']
except KeyError:
    print('No greeter')

if (event['body']) and (event['body'] is not None):
    body = json.loads(event['body'])
    try:
        if (body['greeter']) and (body['greeter'] is not None):
            greeter = body['greeter']
    except KeyError:
        print('No greeter')

res = {
    "statusCode": 200,
    "headers": {
        "Content-Type": "*/*"
    },
    "body": "Hello, " + greeter + "!"
}

return res
```

8. Wählen Sie Deploy (Bereitstellen) aus.

Erstellen Sie eine "Hello, World!"- API

Erstellen Sie jetzt eine API für Ihre "Hello World" Lambda-Funktion unter Verwendung der API Gateway-Konsole.

Erstellen von „Hello, World!“ API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus.

Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Geben Sie in API name (API-Name) **LambdaProxyAPI** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Nachdem Sie eine API erstellt haben, können Sie nun eine Ressource erstellen. In der Regel werden API-Ressourcen in einer Ressourcenstruktur organisiert, die der Anwendungslogik entspricht. Für dieses Beispiel erstellen Sie eine /helloworld-Ressource.

So erstellen Sie eine Ressource

1. Wählen Sie Create Resource (Ressource erstellen) aus.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Ressourcenpfad wird als / beibehalten.
4. Geben Sie für Resource name (Ressourcenname) **helloworld** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

In einer Proxy-Integration wird die gesamte Anfrage unverändert an die Backend-Lambda-Funktion gesendet. Dazu wird eine Catch-All-ANY-Methode verwendet, die eine HTTP-Methode repräsentiert. Der tatsächliche HTTP-Methode wird vom Client zur Laufzeit angegeben. Mit der Methode ANY haben Sie die Möglichkeit, eine einzige API-Methodeneinrichtung für alle unterstützten HTTP-Methoden zu verwenden: DELETE, GET, HEAD, OPTIONS, PATCH, POST und PUT.

Erstellen einer **ANY**-Methode

1. Wählen Sie die /helloworld-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option BELIEBIG aus.

3. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
4. Schalten Sie die Lambda-Proxy-Integration ein.
5. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Funktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
6. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.
7. Wählen Sie Methode erstellen aus.

Bereitstellen und Testen der API

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **test** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.
6. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Browser und cURL verwenden, um eine API mit Lambda-Proxy-Integration zu testen

Sie können einen Browser oder eine [cURL](#) zum Testen Ihrer API verwenden.

Um GET-Anforderungen nur mit Abfragezeichenfolgeparametern zu testen, können Sie die URL der API-helloworld-Ressource in die Adressleiste eines Browsers eingeben.

Fügen Sie die Ressource helloworld und den Abfragezeichenfolgeparameter ?greeter=John an Ihre Aufruf-URL an, um die URL für die API-helloworld-Ressource zu erstellen. Ihre URL sollte wie folgt aussehen.

```
https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?greeter=John
```

Für andere Methoden müssen Sie erweiterte REST-API-Testtools wie [POSTMAN](#) oder [cURL](#) verwenden. In diesem Tutorial wird cURL verwendet. Bei den cURL-Befehl-Beispielen wird davon ausgegangen, dass cURL auf Ihrem Computer installiert ist.

So testen Sie die bereitgestellte API mithilfe von cURL:

1. Öffnen Sie ein Terminal-Fenster.
2. Kopieren Sie den folgenden cURL-Befehl und fügen Sie ihn in das Terminalfenster ein. Ersetzen Sie dann die Aufruf-URL mit der URL, die Sie zuvor kopiert haben und fügen Sie dann **/helloworld** am Ende der URL hinzu.

 Note

Wenn Sie den Befehl unter Windows ausführen, verwenden Sie stattdessen die folgende Syntax:

```
curl -v -X POST "https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld" -H "content-type: application/json" -d "{ \"greeter\": \"John\" }"
```

- a. So rufen Sie die API mit dem Abfragezeichenfolgeparameter `?greeter=John` auf:

```
curl -X GET 'https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?greeter=John'
```

- b. So rufen Sie die API mit einem Header-Parameter von `greeter: John` auf:

```
curl -X GET https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld \
  -H 'content-type: application/json' \
  -H 'greeter: John'
```

- c. So rufen Sie die API mit einem Textkörper von `{"greeter": "John"}` auf:

```
curl -X POST https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld \
  -H 'content-type: application/json' \
  -d '{ "greeter": "John" }'
```

In allen Fällen ist die Ausgabe eine Antwort mit dem folgenden Antworttext: 200

Hello, John!

Tutorial: Erstellen einer REST-API mit einer Lambda-Nicht-Proxy-Integration

In dieser exemplarischen Vorgehensweise verwenden wir die API Gateway-Konsole zur Erstellung einer API, die es einem Client ermöglicht, Lambda-Funktionen über die Lambda Nicht-Proxy-Integration (auch als benutzerdefinierte Integration bekannt) aufzurufen. Weitere Informationen zu AWS Lambda und Lambda-Funktionen finden Sie im [AWS Lambda -Entwicklerhandbuch](#).

Um das Lernen zu erleichtern, haben wir eine einfache Lambda-Funktion mit minimaler API-Einrichtung ausgewählt, um Sie durch die Schritte zur Erstellung einer API Gateway-API mit der benutzerdefinierten Lambda-Integration zu führen. Bei Bedarf werden wir Teile der Logik beschreiben. Ein ausführlicheres Beispiel für die benutzerdefinierte Lambda-Integration finden Sie unter [Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration](#).

Bevor Sie die API erstellen, richten Sie das Lambda-Backend ein, indem Sie eine Lambda-Funktion in AWS Lambda erstellen, wie nachfolgend beschrieben.

Themen

- [Lambda-Funktion für die Nicht-Proxy-Integration von Lambda erstellen](#)
- [API mit Lambda Nicht-Proxy-Integration erstellen](#)
- [Testaufruf der API-Methode](#)
- [Bereitstellen der API](#)
- [Testen der API in einer Bereitstellungsstufe](#)
- [Bereinigen](#)

Lambda-Funktion für die Nicht-Proxy-Integration von Lambda erstellen

Note

Das Erstellen von Lambda-Funktionen kann zu Gebühren für Ihr AWS Konto führen.

In diesem Schritt erstellen Sie eine "Hello World"-ähnliche Lambda-Funktion für die benutzerdefinierte Integration von Lambda. In dieser schrittweisen Anleitung heißt die Funktion `GetStartedLambdaIntegration`.

Die Implementierung dieser `GetStartedLambdaIntegration`-Lambda-Funktion sieht wie folgt aus:

Node.js

```
'use strict';
var days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
  'Saturday'];
var times = ['morning', 'afternoon', 'evening', 'night', 'day'];

export const handler = async(event) => {
  console.log(event);
  // Parse the input for the name, city, time and day property values
  let name = event.name === null || event.name === undefined || event.name === "" ?
'you' : event.name;
  let city = event.city === undefined ? 'World' : event.city;
  let time = times.indexOf(event.time)<0 ? 'day' : event.time;
  let day = days.indexOf(event.day)<0 ? null : event.day;

  // Generate a greeting
  let greeting = 'Good ' + time + ', ' + name + ' of ' + city + '. ';
  if (day) greeting += 'Happy ' + day + '!';

  // Log the greeting to CloudWatch
  console.log('Hello: ', greeting);

  // Return a greeting to the caller
  return {"greeting": greeting}
};
```

Python

```
import json

days = {
    'Sunday',
    'Monday',
    'Tuesday',
    'Wednesday',
```

```

    'Thursday',
    'Friday',
    'Saturday'}
times = {'morning', 'afternoon', 'evening', 'night', 'day'}

def lambda_handler(event, context):
    print(event)
    # parse the input for the name, city, time, and day property values
    name = event.get("name") or 'you'
    city = event.get("city") or 'World'
    try:
        if event['time'] in times:
            time = event['time']
        else:
            time = 'day'
    except KeyError:
        time = 'day'
    try:
        if event['day'] in days:
            day = event['day']
        else:
            day = ''
    except KeyError:
        day = ''
    # Generate a greeting
    greeting = 'Good ' + time + ', ' + name + ' of ' + \
        city + '.' + [' ', ' Happy ' + day + '!'][day != '']
    # Log the greeting to CloudWatch
    print(greeting)

    # Return a greeting to the caller
    return {"greeting": greeting}

```

Für die benutzerdefinierte Lambda-Integration übergibt API Gateway die Eingabe an die Lambda-Funktion vom Client als Integrationsanforderungsdaten. Das event-Objekt des Lambda-Funktionshandlers ist die Eingabe.

Unsere Lambda-Funktion ist einfach. Sie analysiert das event-Objekt mit der Eingabe auf die Eigenschaften name, city, time und day. Sie gibt anschließend ein JSON-Objekt {"message":greeting} an den Aufrufer zurück. Die Nachricht nutzt das "Good [morning|

afternoon|day], [*name*|you] in [*city*|World]. Happy *day*!"-Muster. Es wird angenommen, dass die Eingabe an die Lambda-Funktion das folgenden JSON-Objekt ist:

```
{
  "city": "...",
  "time": "...",
  "day": "...",
  "name" : "..."
```

Weitere Informationen finden Sie im [AWS Lambda -Entwicklerhandbuch](#).

Darüber hinaus protokolliert die Funktion ihre Ausführung bei Amazon, CloudWatch indem sie aufruft `console.log(...)`. Dies ist hilfreich, wenn Sie die Funktionsaufrufe zum Debuggen nachverfolgen möchten. Damit die `GetStartedLambdaIntegration` Funktion den Anruf protokollieren kann, richten Sie eine IAM-Rolle mit entsprechenden Richtlinien für die Lambda-Funktion ein, um die CloudWatch Streams zu erstellen und Protokolleinträge zu den Streams hinzuzufügen. Die Lambda-Konsole führt Sie durch die Erstellung der erforderlichen IAM-Rollen und Richtlinien.

Wenn Sie die API einrichten, ohne die API Gateway-Konsole zu verwenden, wie z. B. beim [Import einer API aus einer OpenAPI-Datei](#), müssen Sie, falls erforderlich, explizit eine Aufrufrolle und Richtlinien für API Gateway erstellen und einrichten, um die Lambda-Funktionen aufzurufen. Weitere Informationen über das Einrichten von Lambda-Aufruf- und Ausführungsrollen für eine API Gateway-API finden Sie unter [Zugriffskontrolle für eine API mit IAM-Berechtigungen](#).

Im Vergleich mit `GetStartedLambdaProxyIntegration`, der Lambda-Proxy-Integration, nimmt die `GetStartedLambdaIntegration` Lambda-Funktion für die benutzerdefinierte Lambda-Integration nur Eingaben aus den API Gateway API-Integrationsanforderungsdaten entgegen. Die Funktion kann eine Ausgabe als beliebiges JSON-Objekt, als Zeichenfolge, als Boolean oder sogar Binär-Blob zurückgeben. Die Lambda-Funktion für die Lambda-Proxy-Integration kann dagegen die Eingabe von beliebigen Anfragedaten entgegennehmen, muss aber ein bestimmtes JSON-Objekt zurückgeben. Die `GetStartedLambdaIntegration`-Funktion für die benutzerdefinierte Lambda-Integration kann API-Anfrageparameter als Eingabe nutzen, sofern API Gateway die erforderlichen API-Anfrageparameter vor dem Weiterleiten der Anfrage an das Backend zum Anfragetext zuordnet. Hierzu muss der API-Entwickler beim Erstellen der API eine Mapping-Vorlage für die API-Methode erstellen und konfigurieren.

Erstellen Sie jetzt die `GetStartedLambdaIntegration` Lambda-Funktion.

So erstellen Sie die **GetStartedLambdaIntegration** Lambda-Funktion für die benutzerdefinierte Integration von Lambda

1. Öffnen Sie die AWS Lambda Konsole unter. <https://console.aws.amazon.com/lambda/>
2. Führen Sie eine der folgenden Aktionen aus:
 - Wenn die Willkommensseite angezeigt wird, wählen Sie Get Started Now (Jetzt beginnen) und dann Create function (Funktion erstellen) aus.
 - Wenn die Listenseite Lambda > Functions (Lambda > Funktionen) angezeigt wird, wählen Sie Create function (Funktion erstellen).
3. Wählen Sie Author from scratch aus.
4. Verfahren Sie im Bereich Author from scratch wie folgt:
 - a. Geben Sie für Name **GetStartedLambdaIntegration** als Namen der Lambda-Funktion ein.
 - b. Wählen Sie für Laufzeit die neueste unterstützte Node.js- oder Python-Laufzeit aus.
 - c. Behalten Sie für Architektur die Standardeinstellung bei.
 - d. Erweitern Sie unter Berechtigungen die Option Standardausführungsrolle ändern. Wählen Sie in der Dropdown-Liste Ausführungsrolle die Option Neue Rolle aus AWS - Richtlinienvorlagen erstellen aus.
 - e. Geben Sie unter Role name (Rollenname) einen Namen für die Rolle ein (z. B. **GetStartedLambdaIntegrationRole**).
 - f. Wählen Sie für Policy templates (Richtlinienvorlagen) die Option Simple microservice permissions (Einfache Microservice-Berechtigungen).
 - g. Wählen Sie Create function (Funktion erstellen).
5. Legen Sie im Bereich Configuration function (Konfiguration) unter Function code (Funktionscode) die Folgendes fest:
 - a. Kopieren Sie den Code in der Lambda-Funktion am Anfang dieses Abschnitts und fügen Sie ihn in den Inline-Codeeditor ein.
 - b. Akzeptieren Sie die Standardwerte für alle anderen Felder in diesem Abschnitt.
 - c. Wählen Sie Deploy (Bereitstellen) aus.
6. Um die neu erstellte Funktion zu testen, wählen Sie die Registerkarte Test aus.
 - a. Geben Sie für Event name (Ereignisname) **HelloWorldTest** ein.

- b. Ersetzen Sie für Ereignis-JSON den Standardcode durch den folgenden Code.

```
{
  "name": "Jonny",
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday"
}
```

- c. Wählen Sie Test, um die Funktion aufzurufen. Der Abschnitt Execution result: succeeded wird angezeigt. Erweitern Sie Details und diese Ausgabe wird angezeigt.

```
{
  "greeting": "Good morning, Jonny of Seattle. Happy Wednesday!"
}
```

Die Ausgabe wird auch in CloudWatch Logs geschrieben.

Als Zusatzaufgabe können Sie die IAM-Konsole verwenden, um die IAM-Rolle (GetStartedLambdaIntegrationRole) anzuzeigen, die bei der Erstellung der Lambda-Funktion erstellt wurde. An diese IAM-Rolle sind zwei Inline-Richtlinien angehängt. Eine legt die grundlegenden Berechtigungen für die Ausführung von Lambda fest. Es ermöglicht das CloudWatch CreateLogGroup Aufrufen aller CloudWatch Ressourcen Ihres Kontos in der Region, in der die Lambda-Funktion erstellt wurde. Diese Richtlinie ermöglicht auch das Erstellen der CloudWatch Streams und das Protokollieren von Ereignissen für die GetStartedLambdaIntegration Lambda-Funktion.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "arn:aws:logs:us-east-1:111111111111:*"
    },
    {
      "Effect": "Allow",
```

```

        "Action": [
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-east-1:111111111111:log-group:/aws/lambda/
GetStartedLambdaIntegration:*"
        ]
    }
}

```

Das andere Richtliniendokument bezieht sich auf das Aufrufen eines anderen AWS Dienstes, der in diesem Beispiel nicht verwendet wird. Sie können sie überspringen.

Der IAM-Rolle ist eine vertrauenswürdige Entität zugeordnet. Diese ist `lambda.amazonaws.com`. Hier ist die Vertrauensstellung:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Die Kombination aus dieser Vertrauensbeziehung und der Inline-Richtlinie ermöglicht es der Lambda-Funktion, eine `console.log()` Funktion aufzurufen, um Ereignisse in Logs zu protokollieren.

CloudWatch

API mit Lambda Nicht-Proxy-Integration erstellen

Wenn die Lambda-Funktion (`GetStartedLambdaIntegration`) erstellt und getestet ist, können Sie die Funktion über eine API Gateway-API zur Verfügung stellen. Zur Veranschaulichung stellen wir die Lambda-Funktion mit einer generischen HTTP-Methode bereit. Wir verwenden den Anforderungstext, eine URL-Pfadvariable, eine Abfragezeichenfolge und einen Header für den Empfang der erforderlichen Eingabedaten vom Client. Wir schalten den API Gateway-Anfragevalidierer für die API ein, um sicherzustellen, dass alle erforderlichen Daten korrekt definiert und spezifiziert sind. Wir konfigurieren eine Zuordnungsvorlage für API Gateway, um die vom Kunden bereitgestellten Anfragedaten in das Format zu transformieren, das von der Backend-Lambda-Funktion benötigt wird.

Eine API mit einer Lambda Nicht-Proxy-Integration erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Geben Sie in API name (API-Name) **LambdaNonProxyAPI** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunktyp bei Regional.
6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Nachdem Sie eine API erstellt haben, können Sie nun eine `/city`-Ressource erstellen. Dies ist ein Beispiel für eine Ressource mit einer Pfadvariablen, die Eingaben vom Client annimmt. In einem späteren Schritt werden Sie diese Pfadvariable der Lambda-Funktionseingabe mittels einer Zuordnungsvorlage zuweisen.

So erstellen Sie eine Ressource

1. Wählen Sie Create Resource (Ressource erstellen) aus.

2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Ressourcenpfad wird als / beibehalten.
4. Geben Sie für Resource name (Ressourcenname) **{city}** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

Nachdem Sie eine `/city`-Ressource erstellt haben, können Sie nun eine ANY-Methode erstellen. Das ANY HTTP-Verb ist ein Platzhalter für eine gültige HTTP-Methode, die ein Client zur Laufzeit sendet. Dieses Beispiel zeigt, dass die ANY-Methode sowohl für die benutzerdefinierte Integration von Lambda als auch für die Integration von Lambda-Proxys verwendet werden kann.

Erstellen einer **ANY**-Methode

1. Wählen Sie die `/city`-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option BELIEBIG aus.
3. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
4. Lassen Sie Lambda proxy integration (Lambda-Proxyintegration) deaktiviert.
5. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Funktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
6. Klicken Sie auf Einstellungen für Methodenanfragen.

Als Nächstes aktivieren Sie eine Anforderungsvalidierung für eine URL-Pfadvariable, einen Abfragezeichenfolgenparameter und einen Header, um sicherzustellen, dass alle erforderlichen Daten definiert sind. Für dieses Beispiel erstellen Sie einen `time`-Abfragezeichenfolgenparameter und einen `day`-Header.

7. Wählen Sie für Anforderungs-Validierer die Option Abfragezeichenfolgeparameter und -Header validieren aus.
8. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
 - b. Geben Sie unter Name **time** ein.
 - c. Aktivieren Sie die Option Erforderlich.
 - d. Caching bleibt ausgeschaltet.

9. Klicken Sie auf HTTP-Anforderungs-Header und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Add header.
 - b. Geben Sie unter Name **day** ein.
 - c. Aktivieren Sie die Option Erforderlich.
 - d. Caching bleibt ausgeschaltet.
10. Wählen Sie Methode erstellen aus.

Nachdem Sie einen Anforderungs-Validator aktiviert haben, konfigurieren Sie die Integrationsanforderung für die ANY-Methode, indem Sie eine Textzuordnungsvorlage hinzufügen, um die eingehende Anfrage in eine JSON-Nutzlast umzuwandeln, wie es die Backend-Lambda-Funktion erfordert.

Konfigurieren der Integrationsanforderung

1. Klicken Sie in der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.
3. Wählen Sie Zuordnungsvorlagen aus.
4. Wählen Sie Add mapping template.
5. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
6. Geben Sie für Vorlagentext den folgenden Code ein:

```
#set($inputRoot = $input.path('$'))
{
  "city": "$input.params('city')",
  "time": "$input.params('time')",
  "day": "$input.params('day')",
  "name": "$inputRoot.callerName"
}
```

7. Wählen Sie Speichern.

Testaufruf der API-Methode

Die API Gateway-Konsole bietet Ihnen eine Testmöglichkeit, mit der Sie den Aufruf der API testen können, bevor sie bereitgestellt wird. Sie verwenden die Testfunktion der Konsole, um die API über das Senden der folgenden Anforderung zu testen:

```
POST /Seattle?time=morning
day:Wednesday

{
  "callerName": "John"
}
```

In dieser Testanforderung legen Sie ANY auf POST fest, {city} auf Seattle, weisen Wednesday als day-Header-Wert zu und weisen "John" als callerName-Wert zu.

So testen Sie die **ANY**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie als Methodentyp die Option POST aus.
3. Geben Sie für Pfad unter Stadt den Wert **Seattle** ein.
4. Für Abfrage-Zeichenketten geben Sie **time=morning** ein.
5. Für Headers geben Sie den Wert **day:Wednesday** ein.
6. Geben Sie für Anforderungstext { **"callerName": "John" }** ein
7. Wählen Sie Test aus.

Überprüfen Sie, ob die zurückgegebene Antwortnutzlast wie folgt lautet:

```
{
  "greeting": "Good morning, John of Seattle. Happy Wednesday!"
}
```

Sie können außerdem die Protokolle anzeigen, um zu untersuchen, wie API Gateway die Anfrage und Antwort verarbeitet.

```
Execution log for request test-request
Thu Aug 31 01:07:25 UTC 2017 : Starting execution for request: test-invoke-request
```

```

Thu Aug 31 01:07:25 UTC 2017 : HTTP Method: POST, Resource Path: /Seattle
Thu Aug 31 01:07:25 UTC 2017 : Method request path: {city=Seattle}
Thu Aug 31 01:07:25 UTC 2017 : Method request query string: {time=morning}
Thu Aug 31 01:07:25 UTC 2017 : Method request headers: {day=Wednesday}
Thu Aug 31 01:07:25 UTC 2017 : Method request body before transformations:
{ "callerName": "John" }
Thu Aug 31 01:07:25 UTC 2017 : Request validation succeeded for content type
application/json
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request URI: https://
lambda.us-west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-
tag=test-request,
Authorization=*****
X-Amz-Date=20170831T010725Z, x-amzn-apigateway-api-id=beags1mnid, X-Amz-
Source-Arn=arn:aws:execute-api:us-west-2:123456789012:beags1mnid/null/POST/
{city}, Accept=application/json, User-Agent=AmazonAPIGateway_beags1mnid,
X-Amz-Security-Token=FQoDYXdzELL////////wEaDMHGzEdEOT/VvGhabiK3AzgKrJw
+3zLqJZG4Ph0q12K6W21+QotY2rrZy0zqhLoiuRg3CAYNQ2eqgL5D54+63ey9bIdtwHGoyBdq8ecWxJK/
YUnT2Rau0L9HCG5p7FC05h3IvwlFfvclQNXeYvskJTLXI05/
yEnY3ttIANpNYL0ezD9Es8rBfyruHfJf0qextKlsC8DymCcqlGkig8qLKcZ0hWJWwiPJiFgL7laabXs+
+ZhCa4hdZo4iqLg729DE4gaV1mJVdoAagIUwLmo+y4NxFDu0r7I0/
E05nYcCrippGVVBYiGk7H4T6sXuhTkbNNqVmXtV3ch5b01h7 [TRUNCATED]
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request body after transformations: {
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday",
  "name" : "John"
}
Thu Aug 31 01:07:25 UTC 2017 : Sending request to https://lambda.us-
west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Received response. Integration latency: 328 ms
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response body before transformations:
{"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response headers: {x-amzn-Remapped-Content-
Length=0, x-amzn-RequestId=c0475a28-8de8-11e7-8d3f-4183da788f0f, Connection=keep-
alive, Content-Length=62, Date=Thu, 31 Aug 2017 01:07:25 GMT, X-Amzn-Trace-
Id=root=1-59a7614d-373151b01b0713127e646635;sampld=0, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Method response body after transformations:
{"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampld=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution

```

```
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200
```

Die Protokolle zeigen die eingehende Anforderung vor der Zuordnung und die Integrationsanforderung nach der Zuordnung. Wenn ein Test fehlschlägt, sind die Protokolle nützlich, um zu prüfen, ob die ursprüngliche Eingabe korrekt ist oder ob die Mapping-Vorlage ordnungsgemäß funktioniert.

Bereitstellen der API

Der Testaufruf ist eine Simulation und hat Einschränkungen. Beispielsweise umgeht er alle Autorisierungsmechanismen für die API. Zum Testen der API-Ausführung in Echtzeit müssen Sie die API bereitstellen. Um eine API bereitzustellen, können Sie einen Snapshot der API erstellen. Der Stufenname definiert den Basispfad entsprechend dem standardmäßigen Hostnamen der API. Die Root-Ressource der API wird nach dem Stufenamen angehängt. Wenn Sie die API ändern, müssen Sie sie erneut für eine neue oder bestehende Stufe bereitstellen, bevor die Änderungen wirksam werden.

So stellen Sie die API für eine Stufe bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **test** ein.

Note

Die Eingabe muss als UTF-8-kodierter (also nicht lokalisierter) Text erfolgen.

4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren. Das allgemeine Muster der Basis-URL lautet `https://api-id.region.amazonaws.com/stageName`. Wenn beispielsweise die Basis-URL der API (beags1mnid) in der Region us-west-2 erstellt und für die Stufe test bereitgestellt wird, lautet der Name `https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Testen der API in einer Bereitstellungsstufe

Es gibt mehrere Möglichkeiten zum Testen einer bereitgestellten API. Für GET-Anforderungen mit nur URL-Pfadvariablen oder Parameter für Abfragezeichenfolgen können Sie die API-Ressourcen-URL in einem Browser eingeben. Für andere Methoden müssen Sie erweiterte REST-API-Testtools wie [POSTMAN](#) oder [cURL](#) verwenden.

So testen Sie die API mithilfe von cURL

1. Öffnen Sie ein Terminalfenster auf Ihrem lokalen Computer, der mit dem Internet verbunden ist.
2. So können Sie dies testen `POST /Seattle?time=evening`:

Kopieren Sie den folgenden cURL-Befehl und fügen Sie ihn in das Terminalfenster ein.

```
curl -v -X POST \
  'https://beags1mnid.execute-api.us-west-2.amazonaws.com/test/Seattle?
time=evening' \
  -H 'content-type: application/json' \
  -H 'day: Thursday' \
  -H 'x-amz-docs-region: us-west-2' \
  -d '{
  "callerName": "John"
}'
```

Als Ergebnis sollten Sie eine erfolgreiche Antwort mit der folgenden Nutzlast erhalten:

```
{"greeting": "Good evening, John of Seattle. Happy Thursday!"}
```

Wenn Sie diese Methode `POST` zu `PUT` ändern, erhalten Sie dieselbe Antwort.

Bereinigen

Wenn Sie die für diese Anleitung erstellten Lambda-Funktionen nicht mehr benötigen, können Sie diese nun löschen. Sie können auch die zugehörigen IAM-Ressourcen löschen.

Warning

Wenn Sie vorhaben, die anderen exemplarischen Vorgehensweisen in dieser Reihe abzuschließen, löschen Sie weder die Lambda-Ausführungsrolle noch die Lambda-Aufruf-Rolle. Wenn Sie eine Lambda-Funktion löschen, auf die Sie APIs sich verlassen, APIs

funktionieren diese nicht mehr. Das Löschen einer Lambda-Funktion kann nicht rückgängig gemacht werden. Wenn Sie die Lambda-Funktion erneut verwenden möchten, müssen Sie die Funktion neu erstellen.

Wenn Sie eine IAM-Ressource löschen, auf die sich eine Lambda-Funktion stützt, funktioniert diese Lambda-Funktion nicht mehr, und alle, APIs die auf diese Funktion angewiesen sind, funktionieren nicht mehr. Das Löschen einer IAM-Ressource kann nicht rückgängig gemacht werden. Wenn Sie die IAM-Ressource wieder nutzen möchten, müssen Sie diese neu erstellen.

So löschen Sie die Lambda-Funktion:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die AWS Lambda Konsole unter. <https://console.aws.amazon.com/lambda/>
2. Wählen Sie in der Liste der Funktionen GetStartedLambdaIntegration, Aktionen und anschließend Delete function. Wählen Sie bei Aufforderung erneut Löschen aus.

Löschen Sie die zugehörigen IAM-Ressourcen wie folgt:

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie in Details Roles.
3. Wählen Sie in der Liste der Rollen GetStartedLambdaIntegrationRole, Role Actions und anschließend Delete role. Führen Sie die Schritte in der Konsole aus, um die Rolle zu löschen.

Tutorial: REST-API mit kontenübergreifender Lambda-Proxy-Integration erstellen

Sie können jetzt eine AWS Lambda Funktion aus einem anderen AWS Konto als Backend für Ihre API-Integration verwenden. Jedes Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist. Dies macht es einfach, Lambda-Backend-Funktionen zentral zu verwalten und für mehrere gemeinsam zu nutzen. APIs

In diesem Abschnitt zeigen wir, wie die kontoübergreifende Lambda-Proxy-Integration mit Hilfe der Amazon API Gateway-Konsole konfiguriert wird.

API für die kontenübergreifende Lambda-Integration mit API Gateway erstellen

So erstellen Sie eine API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Geben Sie in API name (API-Name) **CrossAccountLambdaAPI** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunktyp bei Regional.
6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Lambda-Integrationsfunktion in einem anderen Konto erstellen

Jetzt erstellen Sie eine Lambda-Funktion in einem anderen Konto über das Konto, über das Sie die Beispiel-API erstellt haben.

Erstellen einer Lambda-Funktion in einem anderen Konto

1. Melden Sie sich bei der Lambda-Konsole in einem anderen Konto als dem an, in dem Sie Ihre API Gateway-API erstellt haben.
2. Wählen Sie Create function (Funktion erstellen).
3. Wählen Sie Author from scratch aus.
4. Verfahren Sie unter Author from scratch wie folgt:
 - a. Geben Sie als Function name (Funktionsname) einen Namen ein.
 - b. Wählen Sie aus der Dropdown-Liste Runtime (Laufzeit) eine unterstützte Node.js-Laufzeit aus.
 - c. Behalten Sie für Architektur die Standardeinstellung bei.

- d. Erweitern Sie unter Permissions (Berechtigungen) den Bereich Choose or create an execution role (Ausführungsrolle wählen oder erstellen). Sie können eine Rolle erstellen oder eine vorhandene Rolle auswählen.
- e. Wählen Sie Create function aus, um fortzufahren.
5. Scrollen Sie nach unten bis zum Bereich Function code (Funktionscode).
6. Geben Sie die Node.js-Funktionsimplementierung aus [the section called “Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration”](#) ein.
7. Wählen Sie Deploy (Bereitstellen) aus.
8. Beachten Sie den vollständigen ARN für Ihre Funktion (in der rechten oberen Ecke des Bereichs der Lambda-Funktion). Sie benötigen diese, wenn Sie Ihre kontoübergreifende Lambda-Integration erstellen.

Kontoübergreifende Lambda-Integration konfigurieren

Sobald Sie eine Lambda-Integrationsfunktion in einem anderen Konto haben, können Sie die API Gateway-Konsole verwenden, um sie zu Ihrer API in Ihrem ersten Konto hinzuzufügen.

Note

Wenn Sie einen regionsübergreifenden, kontoübergreifenden Genehmiger konfigurieren, sollte der `sourceArn`, der zur Zielfunktion hinzugefügt wird, die Region der Funktion verwenden, nicht die Region der API.

Nachdem Sie eine API erstellt haben, können Sie nun eine Ressource erstellen. In der Regel werden API-Ressourcen in einer Ressourcenstruktur organisiert, die der Anwendungslogik entspricht. Für dieses Beispiel erstellen Sie eine `/helloworld`-Ressource.

So erstellen Sie eine Ressource

1. Wählen Sie Create Resource (Ressource erstellen) aus.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Ressourcenpfad wird als `/` beibehalten.
4. Geben Sie für Resource name (Ressourcenname) **helloworld** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

Nachdem Sie eine Ressource erstellt haben, können Sie nun eine GET-Methode erstellen. Die GET-Methode wird in einem anderen Konto in der Lambda-Funktion integriert.

So erstellen Sie eine **GET**-Methode

1. Wählen Sie die /helloworld-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
4. Schalten Sie die Lambda-Proxy-Integration ein.
5. Geben Sie für Lambda-Funktion den vollständigen ARN Ihrer Lambda-Funktion aus Schritt 1 ein.

Sie finden den ARN für Ihre Funktion in der Lambda-Konsole in der oberen rechten Ecke des Konsolenfensters.

6. Wenn Sie den ARN eingeben, wird eine `aws lambda add-permission`-Befehlszeichenfolge angezeigt. Diese Richtlinie gewährt Ihrem ersten Konto Zugriff auf die Lambda-Funktion in Ihrem zweiten Konto. Kopieren Sie die `aws lambda add-permission` Befehlszeichenfolge und fügen Sie sie in ein AWS CLI Fenster ein, das für Ihr zweites Konto konfiguriert ist.
7. Wählen Sie Methode erstellen aus.

Die aktualisierte Richtlinie für Ihre Funktion wird in der Lambda-Konsole angezeigt.

(Optional) Anzeige Ihrer aktualisierten Richtlinie

1. Melden Sie sich bei an AWS-Managementkonsole und öffnen Sie die AWS Lambda Konsole unter <https://console.aws.amazon.com/lambda/>.
2. Wählen Sie Ihre Lambda-Funktion aus.
3. Wählen Sie Permissions.

Sie sollten eine Allow-Richtlinie mit einer Condition-Klausel sehen, in der `AWS:SourceArn` die ARN für die GET-Methode Ihrer API ist.

Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels

Sie können die Amazon API Gateway Gateway-Konsole verwenden, um eine einfache REST-API mit der HTTP-Integration für eine PetStore Website zu erstellen und zu testen. Die API-Definition ist als

Swagger-2.0-Datei vorkonfiguriert. Nachdem Sie die API-Definition in API Gateway geladen haben, können Sie die API Gateway-Konsole verwenden, um die Grundstruktur der API zu untersuchen oder die API einfach bereitstellen und testen.

Die PetStore Beispiel-API unterstützt die folgenden Methoden, mit denen ein Client auf die HTTP-Backend-Website von `http://petstore-demo-endpoint.execute-api.com/petstore/pets` zugreifen kann.

 Note

In diesem Tutorial wird ein HTTP-Endpunkt als Beispiel verwendet. Wenn Sie Ihre eigenen erstellen, empfehlen wir Ihnen APIs, HTTPS-Endpunkte für Ihre HTTP-Integrationen zu verwenden.

- GET `/`: Für den Lesezugriff der Root-Ressource der API, die nicht mit jedem Backend-Endpunkt integriert ist. API Gateway antwortet mit einem Überblick über die PetStore Website. Dies ist ein Beispiel für den MOCK-Integrationstyp.
- GET `/pets`: Für den Lesezugriff auf die `/pets`-Ressource der API, die mit der gleichnamigen Backend-`/pets`-Ressource integriert ist. Das Backend gibt eine Seite mit verfügbaren Haustieren in der PetStore zurück. Dies ist ein Beispiel für den HTTP-Integrationstyp. Die URL des Integrationsendpunktes lautet `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- POST `/pets`: Für den Schreibzugriff auf die `/pets`-Ressource der API, die mit der Backend-`/petstore/pets`-Ressource integriert ist. Nach Erhalt einer korrekten Anfrage fügt das Backend das angegebene Haustier zur Liste hinzu PetStore und gibt das Ergebnis an den Anrufer zurück. Die Integration ist ebenfalls HTTP.
- GET `/pets/{petId}`: Für den Lesezugriff auf ein Haustier, das durch einen `petId`-Wert per Pfadvariable der URL der eingehenden Anforderung identifiziert wird. Diese Methode hat ebenfalls den HTTP-Integrationstyp. Das Backend gibt das angegebene Haustier zurück, das in der gefunden wurde. PetStore Die URL des Backend-HTTP-Endpunkts ist `http://petstore-demo-endpoint.execute-api.com/petstore/pets/n`, wobei *n* eine Ganzzahl für die ID des abgefragten Haustiers ist.

Die API unterstützt den CORS-Zugriff über die `OPTIONS`-Methoden des `MOCK`-Integrationstyps. API Gateway gibt die erforderlichen Header zurück, die den CORS-Zugriff unterstützen.

Das folgende Verfahren führt Sie durch die Schritte zum Erstellen und Testen einer API anhand eines Beispiels mit der API Gateway-Konsole.

So importieren, erstellen und testen Sie die Beispiel-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Führen Sie eine der folgenden Aktionen aus:
 - Um Ihre erste API zu erstellen, wählen Sie für REST-API die Option Erstellen.
 - Wenn Sie zuvor eine API erstellt haben, wählen Sie API erstellen und dann für REST-API die Option Erstellen.
3. Wählen Sie unter Create REST API (REST API erstellen) die Option Example API (Beispiel-API) und dann Create API (API erstellen) aus, um die Beispiel-API zu erstellen.

Create REST API

API details

☐ New API
Create a new REST API.

☐ Clone existing API
Create a copy of an API in this AWS account.

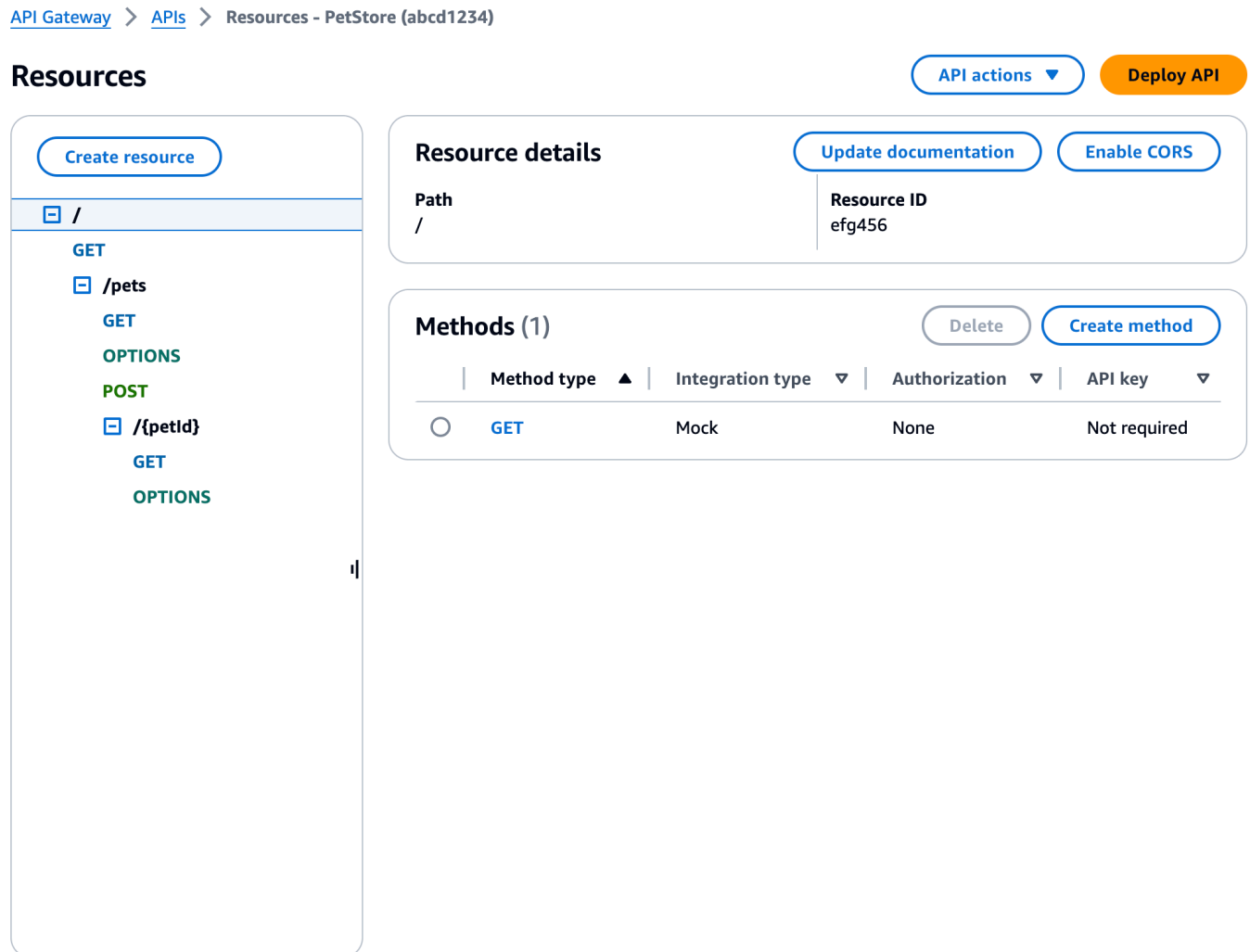
☐ Import API
Import an API from an OpenAPI definition.

☒ Example API
Learn about API Gateway with an example API.

```
1 {
2   "swagger": "2.0",
3   "info": {
4     "description": "Your first API with Amazon API Gateway. This is a sample API that integrates via HTTP w
ith our demo Pet Store endpoints",
5     "title": "PetStore"
6   },
7   "schemes": [
8     "https"
9   ],
10  "paths": {
11    "/": {
12      "get": {
13        "tags": [
14          "pets"
15        ],
16        "description": "PetStore HTML web page containing API usage information",
17        "consumes": [
18          "application/json"
19        ],
```

Sie können einen Bildlauf durch die OpenAPI-Definition ausführen und Details zu dieser Beispiel-API anzeigen. Klicken Sie anschließend auf Create API (API erstellen).

4. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen). Die folgende Abbildung zeigt die neu erstellte API:



Im Bereich Resources wird die Struktur der erstellten API als Knotenstruktur gezeigt. Die API-Methoden der einzelnen Ressourcen werden als Edges in der Struktur dargestellt. Bei Auswahl einer Ressource werden alle zugehörigen Methoden auf der rechten Seite in der Tabelle Methods (Methoden) angegeben. Bei jeder Methode werden der Methodentyp, der Integrationstyp, der Autorisierungstyp und die API-Schlüsselanforderung angezeigt.

- Um die Details einer Methode anzuzeigen, die Einrichtung zu ändern oder den Methodenaufruf zu testen, wählen Sie entweder in der Methodenliste oder der Ressourcenstruktur einen Methodennamen aus. Zur Illustration wählen wir hier die Methode `POST /pets`:

Resources

API actions ▾

Deploy API

Create resource

/

GET

/pets

GET

OPTIONS

POST

/petid

GET

OPTIONS

Update documentation

Delete

/pets - POST - Method execution

ARN

arn:aws:execute-api:us-west-1:111222333:abcd1234/*/POST/pets

Resource ID

aaa111

Client

Method request

Integration request

HTTP

HTTP integration

Method response

Integration response

Method request

Integration request

Integration response

Method response

Method request settings

Authorization

NONE

Request validator

None

API key required

False

SDK operation name

CreatePet

Request paths (0)

Name

Caching

Der daraufhin angezeigte Bereich zur Methodenausführung bietet eine logische Ansicht der Struktur und des Verhaltens der ausgewählten (POST /pets) Methode.

Die Method request (Methodenanforderung) und die Method response (Methodenantwort) stellen die Schnittstelle der API zum Frontend dar, und die Integration request (Integrationsanforderung) und die Integration response (Integrationsantwort) stellen die Schnittstelle der API zum Backend dar.

Ein Client verwendet die API für den Zugriff auf eine Backend-Funktion über Method Request (Methodenanforderung). Bei Bedarf wandelt Amazon API Gateway die Clientanforderung in ein Format um, das vom Backend in Integration Request (Integrationsanfrage) verarbeitet werden kann, bevor die eingehende Anfrage an das Backend weitergeleitet wird. Die umgewandelte Anforderung wird als Integrationsanforderung bezeichnet. Analog dazu gibt das Backend die Antwort an API Gateway in Integration Response (Integrationsantwort) zurück. API Gateway

Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels

62

leitet dann an Method Response (Methodenantwort) weiter, bevor das Senden an den Client erfolgt. Auch hier kann API Gateway, falls erforderlich, die Backend-Antwortdaten einer vom Client erwarteten Form zuordnen.

Für die POST-Methode einer API-Ressource kann die Nutzlast der Methodenanforderung ohne Änderung an die Integrationsanforderung übergeben werden, wenn die Nutzlast der Methodenanforderung dasselbe Format wie die Nutzlast der Integrationsanforderung aufweist.

Die GET /-Methodenanforderung verwendet den Integrationstyp MOCK und ist nicht mit einem tatsächlichen Endpunkt im Backend verknüpft. Die entsprechende Integration Response (Integrationsantwort) ist so eingerichtet, dass sie eine statische HTML-Seite zurückgibt: Wenn die Methode aufgerufen wird, akzeptiert API Gateway einfach die Anforderung und gibt die konfigurierte Integrationsantwort sofort über Method Response (Methodenantwort) an den Client zurück. Mit der Pseudo-Integration können Sie eine API testen, ohne dass ein Backend-Endpunkt erforderlich ist. Sie können sie auch zur Bereitstellung einer lokalen Antwort verwenden, die aufgrund einer Antwort-TextMapping-Vorlage generiert wurde.

Als API-Entwickler steuern Sie das Frontend-Interaktionsverhalten der API, indem Sie eine Methodenanforderung und eine Methodenantwort konfigurieren. Das Backend-Interaktionsverhalten der API steuern Sie, indem Sie eine Integrationsanforderung und eine Integrationsantwort einrichten. Dafür sind Datenzuweisungen zwischen einer Methode und der entsprechenden Integration nötig. Im Moment konzentrieren wir uns darauf, die API zu testen, um eine end-to-end Benutzererfahrung zu bieten.

6. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
7. Geben Sie z. B. zum Testen der POST /pets-Methode die folgende **{"type": "dog", "price": 249.99}**-Nutzlast in Request Body (Anforderungstext) ein, bevor Sie die Schaltfläche Test betätigen.



Integration request

Integration response

Method response

Test

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Query strings

```
param1=value1&param2=value2
```

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

```
header1:value1  
header2:value2
```

Client certificate

No client certificates have been generated.

Request body

```
1 {  
2   "type" : "dog", "price": 249.99  
3 }
```

Die Eingabe spezifiziert die Attribute des Haustiers, das wir der Liste der Haustiere auf der PetStore Website hinzufügen möchten.

- Die Ergebnisse werden folgendermaßen angezeigt:

/pets - POST method test results**Request**`/pets`**Latency ms**`13`**Status**`200`**Response body**

```
{
  "pet": {
    "type": "dog",
    "price": 249.99
  },
  "message": "success"
}
```

Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-67928ab3-ade6b7bdc102614983b74a5f"
}
```

Logs

Execution log for request 06089002-b249-47e9-b485-5530e4843290

Thu Jan 23 18:30:11 UTC 2025 : Starting execution for request: 06089002-b249-47e9-b485-

In der Ausgabe werden unter Log (Protokoll) die Statusänderungen von der Methodenanforderung zur Integrationsanforderung und von der Integrationsantwort zur Methodenantwort angezeigt. Das ist hilfreich, um Zuweisungsfehler zu beheben, die zu einer fehlerhaften Anforderung führen. In diesem Beispiel wird keine Zuweisung angewendet: Die Nutzlast der Methodenanforderung wird über die Integrationsanforderung an das Backend übergeben und die Backend-Antwort wird parallel dazu über die Integrationsantwort an die Methodenantwort übergeben.

Um die API mit einem anderen Client als der API Gateway test-invoke-request Gateway-Funktion zu testen, müssen Sie die API zunächst in einer Phase bereitstellen.

9. Wählen Sie Deploy API (API bereitstellen) aus, um die Beispiel-API bereitzustellen.

[API Gateway](#) > [APIs](#) > Resources - PetStore (abcd1234)

Resources

API actions ▼

Deploy API

Create resource



/

GET

/pets

GET

OPTIONS

POST

/{petId}

GET

OPTIONS

Resource details

Path

/

Update documentation

Enable CORS

Resource ID

efg456

Methods (1)

Delete

Create method

	Method type ▲	Integration type ▼	Authorization ▼	API key ▼
☐	GET	Mock	None	Not required

10. Wählen Sie für Stage (Stufe) die Option New stage (Neue Stufe) aus, und geben Sie dann **test** ein.
11. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
12. Wählen Sie Bereitstellen.
13. Im resultierenden Bereich Stages (Stufen) zeigt Invoke URL (Aufruf-URL) unter Stage details (Stufendetails) die URL an, die die GET /-Methodenanfrage der API aufruft.

Stage details [Info](#) [Edit](#)

Stage name Prod	Rate Info 10000	Web ACL -
Cache cluster Info ⊖ Inactive	Burst Info 5000	Client certificate -
Default method-level caching ⊖ Inactive		

Invoke URL
 `https://abcd1234.execute-api.us-east-1.amazonaws.com/Prod`

Active deployment
efg456 on June 01, 2023, 15:00 (UTC-07:00)

- Wählen Sie das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Eine erfolgreiche Antwort gibt das Ergebnis zurück, das von der Mapping-Vorlage in der Integrationsantwort generiert wurde.
- Erweitern Sie im Navigationsbereich Stufen die Stufe Test, wählen Sie GET für `/pets/{petId}` aus und kopieren Sie dann den Aufruf-URL-Wert `https://api-id.execute-api.region.amazonaws.com/test/pets/{petId}`. `{petId}` steht für eine Pfadvariable.

Fügen Sie den (im vorigen Schritt kopierten) Invoke URL-Wert in die Adresszeile des Browsers ein und ersetzen Sie `{petId}` durch z. B. 1. Drücken Sie zum Senden der Anforderung die Eingabetaste. Als Ergebnis wird die Antwort "200 OK" mit der folgenden JSON-Nutzlast zurückgegeben:

```
{
  "id": 1,
  "type": "dog",
  "price": 249.99
}
```

Die API-Methode kann auf diese Weise aufgerufen werden, da der Authorization-Typ auf NONE festgelegt ist. Falls die AWS_IAM-Autorisierung verwendet wird, muss die Anfrage mit [Signature Version 4](#) (SigV4)- oder [Signature Version 4a](#) (SigV4a)-Protokollen signiert werden. Ein Beispiel für eine solche Anforderung finden Sie unter [the section called "Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen"](#).

Wählen Sie ein Tutorial zur HTTP-Integration aus

Sie können entweder eine HTTP-Proxy-Integration oder eine benutzerdefinierte HTTP-Integration verwenden, um eine API mit HTTP-Integration zu erstellen.

Bei einer HTTP-Proxy-Integration müssen Sie lediglich die HTTP-Methode und die HTTP-Endpunkt-URI entsprechend der Backend-Anforderungen festlegen. Wir empfehlen, die HTTP-Proxy-Integration zu verwenden (sofern möglich), um die Vorteile der optimierten API-Einrichtung zu genießen.

Die benutzerdefinierte HTTP-Integration kann die bessere Wahl sein, wenn Client-Anforderungsdaten für das Backend oder Backend-Antwortdaten für den Client transformiert werden müssen.

Themen

- [Tutorial: Erstellen einer REST-API mit HTTP-Proxy-Integration](#)
- [Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen](#)

Tutorial: Erstellen einer REST-API mit HTTP-Proxy-Integration

Die HTTP-Proxy-Integration ist ein einfacher, leistungsstarker und vielseitiger Mechanismus für den Aufbau einer API, mit der eine Webanwendung auf mehrere Ressourcen oder Funktionen des integrierten HTTP-Endpunktes (z. B. die gesamte Website) zugreifen kann und der eine unkomplizierte Einrichtung einer einzigen API-Methode verwendet. Bei der HTTP-Proxy-Integration leitet API Gateway die vom Client übermittelte Methodenanforderung an das Backend weiter. Diese übergebenen Anforderungsdaten umfassen die Abfrage-Header, Parameter für Abfragezeichenfolgen, die URL-Pfadvariablen und die Nutzlast. Der Backend-HTTP-Endpunkt oder Webserver analysiert die eingehenden Anforderungsdaten, um die eigene Antwort zu bestimmen. Die HTTP-Proxy-Integration sorgt dafür, dass der Client und das Backend direkt interagieren, ohne dass nach dem Einrichten der API-Methode ein Eingriff von API Gateway erfolgt. Dies gilt nicht für bekannte Probleme wie nicht unterstützte Zeichen, die in [the section called "Wichtige Hinweise"](#) aufgeführt sind.

Mit der umfassenden Proxy-Ressource {proxy+} und dem umfassenden ANY-Verb für die HTTP-Methode können Sie mit einer HTTP-Proxy-Integration eine API mit einer einzelnen API-Methode erstellen. Die Methode macht den gesamten Satz der öffentlich verfügbaren HTTP-Ressourcen und -Vorgänge der Website verfügbar. Wenn der Backend-Webserver mehr Ressourcen für den öffentlichen Zugriff öffnet, kann der Client diese neuen Ressourcen mit derselben API-Einrichtung nutzen. Um dies zu ermöglichen, muss der Websiteentwickler dem Cliententwickler

klar kommunizieren, was die neuen Ressourcen sind und welche Vorgänge für diese Umgebungen anwendbar sind.

Als kurze Einführung zeigt die folgende Anleitung die HTTP-Proxy-Integration. Im Tutorial erstellen wir mithilfe der API Gateway-Konsole eine API, um sie über eine generische Proxyressource in die PetStore Website zu integrieren{proxy+}, und erstellen den Platzhalter für die HTTP-Methode. ANY

Themen

- [API mit HTTP-Proxy-Integration unter Verwendung der API Gateway-Konsole erstellen](#)
- [Testen einer API mit HTTP-Proxy-Integration](#)

API mit HTTP-Proxy-Integration unter Verwendung der API Gateway-Konsole erstellen

Das folgende Verfahren führt Sie durch die Schritte zum Erstellen und Testen einer API mit einer Proxy-Ressource für ein HTTP-Backend unter Verwendung der API Gateway-Konsole. Das HTTP-Backend ist die PetStore-Website (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>) von [Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen](#), auf der Screenshots als visuelle Hilfsmittel zur Veranschaulichung der API Gateway-Benutzeroberflächenelemente verwendet werden. Wenn Ihnen die Verwendung der API Gateway-Konsole zur Erstellung einer API unbekannt ist, sollten Sie zuerst diesen Abschnitt lesen.

So erstellen Sie eine API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Geben Sie in API name (API-Name) **HTTPProxyAPI** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunktyp bei Regional.
6. Wählen Sie als IP-Adresstyp. IPv4

7. Wählen Sie Create API (API erstellen) aus.

In diesem Schritt erstellen Sie einen Proxy-Ressourcenpfad für `{proxy+}`. Dies ist der Platzhalter für beliebige Backend-Endpunkte unter `http://petstore-demo-endpoint.execute-api.com/`. Zum Beispiel kann dies `petstore`, `petstore/pets` und `petstore/pets/{petId}` sein. API Gateway erstellt die ANY-Methode, wenn Sie die `{proxy+}`-Ressource erstellen, und dient als Platzhalter für alle unterstützten HTTP-Verben zur Laufzeit.

So erstellen Sie eine `/ {proxy+}`-Ressource

1. Wählen Sie Ihre API aus.
2. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
3. Wählen Sie Create Resource (Ressource erstellen) aus.
4. Aktivieren Sie die Option Proxy-Ressource.
5. Ressourcenpfad wird als `/` beibehalten.
6. Geben Sie für Resource name (Ressourcenname) **`{proxy+}`** ein.
7. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
8. Wählen Sie Create Resource (Ressource erstellen) aus.

Create resource

Resource details

☒ **Proxy resource** [Info](#)
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path / {proxy+} / **Resource name** {proxy+}

☐ **CORS (Cross Origin Resource Sharing)** [Info](#)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#) [Create resource](#)

In diesem Schritt integrieren Sie die ANY-Methode mithilfe einer Proxyintegration in einen Backend-HTTP-Endpunkt. Bei einer Proxy-Integration leitet API Gateway die vom Client übermittelte Methodenanforderung ohne Zutun der API-Gateway an das Backend weiter.

Erstellen einer **ANY**-Methode

1. Wählen Sie die Ressource `/ {proxy+} /` aus.
2. Wählen Sie die Methode BELIEBIGE.
3. Klicken Sie unter dem Warnsymbol auf Integration bearbeiten. Sie können keine API mit einer Methode bereitstellen, die keine Integration hat.
4. Wählen Sie für den Integrationstyp die Option HTTP aus.
5. Aktivieren Sie die HTTP-Proxy-Integration.
6. Für HTTP-Methode wählen Sie BELIEBIGE aus.
7. Geben Sie für Endpunkt-URL `http://petstore-demo-endpoint.execute-api.com/ {proxy}` ein.
8. Wählen Sie Speichern.

Testen einer API mit HTTP-Proxy-Integration

Ob eine bestimmte Clientanforderung erfolgreich ist, hängt von Folgendem ab:

- Ob das Backend den entsprechenden Backend-Endpunkt verfügbar gemacht hat und, falls dies der Fall ist, ob die erforderlichen Zugriffsberechtigungen erteilt wurden.
- Ob der Client die korrekte Eingabe bereitstellt.

Die hier verwendete PetStore API macht die `/petstore` Ressource beispielsweise nicht verfügbar. Daher erhalten Sie eine `404 Resource Not Found`-Antwort mit der Fehlermeldung `Cannot GET /petstore`.

Darüber hinaus muss der Client in der Lage sein, das Ausgabeformat zu verarbeiten, um das Ergebnis des Backends korrekt zu analysieren. API Gateway fungiert nicht als Vermittler zur Vereinfachung der Interaktionen zwischen dem Client und dem Backend.

Um eine in die PetStore Website integrierte API mithilfe der HTTP-Proxyintegration über die Proxy-Ressource zu testen

1. Wählen Sie die Registerkarte **Test**. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie als Methodentyp die Option **GET** aus.
3. Geben Sie für Pfad unter Proxy **petstore/pets** ein.
4. Für Abfragezeichenketten geben Sie **type=fish** ein.
5. Wählen Sie **Test** aus.

[<](#) [Test](#) [Integration request](#) [Integration response](#) [Method response](#) [Test](#) [>](#)

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Method type

GET ▼

Path

proxy

petstore/pets

Query strings

type=fish

Da die Backend-Website die GET /petstore/pets?type=fish-Anforderung unterstützt, wird eine erfolgreiche Antwort wie die folgende zurückgegeben:

```
[
  {
    "id": 1,
    "type": "fish",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "fish",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```


Wenn Sie versuchen, GET /petstore aufzurufen, erhalten Sie eine 404-Antwort mit der Fehlermeldung Cannot GET /petstore. Der Grund hierfür ist, dass das Backend die angegebene Operation nicht unterstützt. Wenn Sie anrufen GET /petstore/pets/1, erhalten Sie eine 200 OK Antwort mit der folgenden Nutzlast, da die Anfrage von der PetStore Website unterstützt wird.

```
{
  "id": 1,
  "type": "dog",
  "price": 249.99
}
```

Sie können Ihre API zusätzlich in einem Browser testen. Stellen Sie Ihre API bereit und ordnen Sie sie einer Stufe zu, um die Aufruf-URL Ihrer API zu erstellen.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **test** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

Jetzt können Clients Ihre API aufrufen.

So rufen Sie Ihre API auf

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Klicken Sie im Hauptnavigationsbereich auf Stufe.
4. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Geben Sie die Aufruf-URL Ihrer API in einen Webbrowser ein.

Die URL sollte wie folgt aussehen: `https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

Ihr Browser sendet eine GET-Anforderung an die API.

5. Das Ergebnis sollte mit dem übereinstimmen, was Ihr Test in der API Gateway-Konsole zurückgegeben hat.

Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen

In diesem Tutorial erstellen Sie mit der Amazon API Gateway-Konsole eine API von Grund auf. Im Prinzip nutzen Sie die Konsole als API-Designstudio und wählen damit die API-Funktionen aus, testen das API-Verhalten, erstellen die API und stellen sie in Stufen bereit.

Themen

- [Erstellen einer API mit benutzerdefinierter HTTP-Integration](#)
- [\(Optionale\) Parameter für die Zuordnungsanforderung](#)

Erstellen einer API mit benutzerdefinierter HTTP-Integration

In diesem Abschnitt werden die Schritte beschrieben, mit denen Sie Ressourcen erstellen, Methoden für eine Ressource nutzen, eine Methode für bestimmte API-Verhaltensweisen konfigurieren und die API testen und bereitstellen.

In diesem Schritt erstellen Sie eine leere API. In den folgenden Schritten erstellen Sie Ressourcen und Methoden, um Ihre API mithilfe einer Nicht-Proxy-HTTP-Integration mit dem `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-Endpunkt zu verbinden.

So erstellen Sie eine API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie **Create API** (API erstellen). Wählen Sie unter **REST-API** die Option **Build** (Erstellen) aus.

3. Geben Sie in **API name** (API-Name) **HTTPNonProxyAPI** ein.
4. (Optional) Geben Sie unter **Description** (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für **API-Endpointtyp** bei **Regional**.
6. Wählen Sie als **IP-Adresstyp** die Option. **IPv4**
7. Wählen Sie **Create API** (API erstellen) aus.

In der **Resources**-Struktur wird die Stammressource (/) ohne Methoden angezeigt. In dieser Übung erstellen wir die API mit der benutzerdefinierten HTTP-Integration der PetStore Website (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>.) Zur Veranschaulichung erstellen wir eine **/pets** Ressource als untergeordnetes Element des Stammverzeichnis und stellen eine **GET**-Methode für diese Ressource bereit, damit ein Client eine Liste verfügbarer Pets-Elemente von der PetStore Website abrufen kann.

Erstellen einer /pets-Ressource

1. Wählen Sie **Create Resource** (Ressource erstellen) aus.
2. Die **Proxy-Ressource** bleibt ausgeschaltet.
3. **Ressourcenpfad** wird als **/** beibehalten.
4. Geben Sie für **Resource name** (Ressourcenname) **pets** ein.
5. **CORS** (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie **Create Resource** (Ressource erstellen) aus.

In diesem Schritt erstellen Sie eine **GET**-Methode für die **/pets**-Ressource. Die **GET**-Methode ist in die <http://petstore-demo-endpoint.execute-api.com/petstore/pets>-Website integriert. Die Folgenden sind weitere Optionen für eine API-Methode:

- **POST**; wird vor allem für die Erstellung von untergeordneten Ressourcen verwendet.
- **PUT**; wird vor allem für die Aktualisierung vorhandener Ressourcen verwendet (kann auch für die Erstellung untergeordneter Ressourcen verwendet werden, obwohl dies nicht empfohlen wird).
- **DELETE**; wird für das Löschen von Ressourcen verwendet.
- **PATCH**; wird für die Aktualisierung von Ressourcen verwendet.

- HEAD; wird vor allem in Testszenarien verwendet. Diese Option ist identisch mit GET, gibt aber nicht die Repräsentation der Ressource zurück.
- OPTIONS – kann von Aufrufern verwendet werden, um Informationen über verfügbare Kommunikationsoptionen für den Zielservice abzurufen.

Als HTTP method der Integrationsanforderung müssen Sie eine Methode wählen, die vom Backend unterstützt wird. Bei HTTP oder Mock integration ist es durchaus sinnvoll, dass die Methoden- und die Integrationsanforderung das gleiche HTTP-Verb nutzen. Bei den anderen Integrationstypen verwenden Methoden- und Integrationsanforderung vermutlich unterschiedliche HTTP-Verben. Um beispielsweise eine Lambda-Funktion aufzurufen, muss die Integrationsanforderung POST verwenden, wohingegen die Methodenanforderung – abhängig von der Logik der Lambda-Funktion – ein beliebiges HTTP-Verb nutzen kann.

So erstellen Sie eine **GET**-Methode für die /pets-Ressource

1. Wählen Sie die /pets-Ressource aus.
2. Wählen Sie Methode erstellen aus.
3. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
4. Für den Integrationstyp wählen Sie die Option HTTP-Integration aus.
5. Die Option Lambda-Proxy-Integration bleibt deaktiviert.
6. Für HTTP-Methode wählen Sie GET aus.
7. Geben Sie für Endpunkt-URL **`http://petstore-demo-endpoint.execute-api.com/petstore/pets`** ein.

Auf der PetStore Website können Sie auf einer bestimmten Seite eine Liste von Pet Objekten abrufen, die nach der Art des Haustiers, z. B. „Hund“ oder „Katze“, sortiert sind.

8. Für Inhaltsbehandlung wählen Sie Passthrough aus.
9. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

Die PetStore Website verwendet die Parameter type und die page Abfragezeichenfolge, um eine Eingabe zu akzeptieren. Hierfür müssen diese Abfragezeichenfolge-Parameter zur Methodenanforderung hinzugefügt, und den entsprechenden Abfragezeichenfolgen der Integrationsanforderung zugewiesen werden.

10. Fügen Sie die Abfragezeichenfolge-Parameter wie folgt hinzu:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

- b. Für Name geben Sie **type** ein.
- c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.

Wiederholen Sie die obigen Schritte, um eine zusätzliche Abfragezeichenfolge namens **page** zu erstellen.

11. Wählen Sie Methode erstellen aus.

Bei der Anforderungsübermittlung kann der Client nun einen Haustiertyp und eine Seitenzahl als Abfragezeichenfolge-Parameter senden. Diese Eingabeparameter müssen den Abfragezeichenfolgenparametern der Integration zugeordnet werden, um die Eingabewerte an unsere PetStore Website im Backend weiterzuleiten.

Zuordnen von Eingabeparametern zur Integrationsanforderung

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Abfragezeichenfolgen-Parameter und gehen Sie dann wie folgt vor:
 - a. Klicken Sie auf Abfragezeichenfolge-Parameter hinzufügen.
 - b. Geben Sie unter Name **type** ein.
 - c. Für Zugeordnet von geben Sie **method.request.querystring.type** ein.
 - d. Caching bleibt ausgeschaltet.
 - e. Klicken Sie auf Abfragezeichenfolge-Parameter hinzufügen.
 - f. Geben Sie unter Name **page** ein.
 - g. Für Zugeordnet von geben Sie **method.request.querystring.page** ein.
 - h. Caching bleibt ausgeschaltet.
3. Wählen Sie Speichern.

So testen Sie die API-Daten

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Abfrage-Zeichenketten geben Sie **type=Dog&page=2** ein.
3. Wählen Sie Test aus.

Das Ergebnis sollte wie folgt aussehen:

Method request

Integration request

Integration response

Method response

Test

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Query strings

type=Dog&page=2

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

header1:value1
header2:value2

Client certificate

None ▼

Test

i /pets - GET method test results

Request	Latency ms	Status
/pets?type=Dog&page=2	10	200

Response body

[
 {
 "id": 4,
 "type": "Dog",
 "price": 999.99
 }]

Nach erfolgreich ausgeführtem Test kann die API bereitgestellt und damit öffentlich verfügbar gemacht werden.

4. Klicken Sie auf Deploy API.
5. Wählen Sie für Stufe die Option Neue Stufe aus.
6. Geben Sie für Stage name (Stufenname) **Prod** ein.
7. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
8. Wählen Sie Bereitstellen.

9. (Optional) Klicken Sie unter Stufendetails auf das Kopiersymbol für Aufruf-URL, um die URL Ihrer Aufruf-API zu kopieren. Sie können diese mit Tools wie [Postman](#) und [cURL](#) zum Testen Ihrer API verwenden.

Wenn Sie ein SDK zum Erstellen eines Clients verwenden, können Sie die Anforderung mithilfe der vom SDK bereitgestellten Methodenaufrufe signieren. Weitere Informationen zur Implementierung finden Sie jeweils im ausgewählten [AWS SDK](#).

 Note

Nach jeder Änderung der API müssen Sie die API erneut bereitstellen, um die neuen oder aktualisierten Funktionen verfügbar zu machen, bevor die Anforderungs-URL wieder aufgerufen wird.

(Optionale) Parameter für die Zuordnungsanforderung

Anfrageparameter für ein API Gateway API zuordnen

In diesem Tutorial erfahren Sie, wie Sie einen {petId}-Pfadparameter über die Methodenanforderung der API erstellen, um eine Element-ID vorzugeben, diese dem {id}-Pfadparameter in der Integrationsanforderungs-URL zuweisen, und dann die Anforderung an den HTTP-Endpunkt senden.

 Note

Eine Verwechslung von Groß- und Kleinschreibung kann im späteren Verlauf zu Fehlern führen.

Schritt 1: Erstellen von Ressourcen

In diesem Schritt erstellen Sie eine Ressource mit einem {petId}-Pfadparameter.

Erstellen der {petId}-Ressource

1. Wählen Sie die /pets-Ressource aus und klicken Sie dann auf Ressource erstellen.
2. Die Proxy-Ressource bleibt ausgeschaltet.

3. Für Ressourcenpfad wählen Sie die Option/pets/aus.
4. Geben Sie für Resource name (Ressourcenname) **{petId}** ein.

Setzen Sie geschweifte Klammern ({ }) um petId, sodass /pets/{petId} angezeigt wird.

5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

Schritt 2: Erstellen und Testen der Methoden

In diesem Schritt erstellen Sie eine GET-Methode mit einem {petId}-Pfadparameter.

Einrichten der GET-Methode

1. Wählen Sie die /{petId}-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Für den Integrationstyp wählen Sie die Option HTTP-Integration aus.
4. Die Option Lambda-Proxy-Integration bleibt deaktiviert.
5. Für HTTP-Methode wählen Sie GET aus.
6. Für Endpunkt-URL geben Sie **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}** ein.
7. Für Inhaltsbehandlung wählen Sie Passthrough aus.
8. Das Standard-Timeout bleibt aktiviert.
9. Wählen Sie Methode erstellen aus.

Als Nächstes ordnen Sie den soeben erstellten {petId}-Pfadparameter dem {id}-Pfadparameter in der HTTP-Endpunkt-URL der Integrationsanforderung zu. Die HTTP-Endpunkt-URL lautet **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}**.

Zuordnen des **{petId}**-Pfadparameters

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Pfadparameter.
3. API Gateway erstellt einen Pfadparameter für die Integrationsanforderung namens petId. Dieser Pfadparameter ist jedoch nicht für die HTTP-Endpunkt-URL gültig, die Sie als Backend-

Integration festgelegt haben. Der HTTP-Endpunkt verwendet `{id}` als Pfadparameter. Unter Name löschen Sie also `petId` und geben statt dessen **`id`** ein.

Damit wird der Pfadparameter `petId` der Methodenanforderung dem Pfadparameter `id` der Integrationsanforderung zugewiesen.

4. Wählen Sie Speichern.

Jetzt können Sie die Methode testen.

So testen Sie die -Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Geben Sie unter Pfad für `petId` den Wert **4** ein.
3. Wählen Sie Test aus.

Wenn diese Aktion erfolgreich ist, zeigt Antworttext Folgendes an:

```
{
  "id": 4,
  "type": "bird",
  "price": 999.99
}
```

Schritt 3: Bereitstellen der API

In diesem Schritt stellen Sie die API bereit, sodass Sie beginnen können, sie außerhalb der API-Gateway-Konsole aufzurufen.

So stellen Sie die API bereit

1. Klicken Sie auf Deploy API.
2. Für Stufe wählen Sie die Option Prod aus.
3. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
4. Wählen Sie Bereitstellen aus.

Schritt 4: Testen der API

In diesem Schritt wechseln Sie zu der API Gateway-Konsole und verwenden Ihre API, um auf den HTTP-Endpunkt zuzugreifen.

1. Klicken Sie im Hauptnavigationsbereich auf Stufe.
2. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Sie sollte wie folgt aussehen:

```
https://my-api-id.execute-api.region-id.amazonaws.com/prod
```

3. Geben Sie diese URL in das Adressfeld einer neuen Registerkarte im Browser ein und fügen Sie `/pets/4` an die URL an, bevor Sie Ihre Anforderung absenden.
4. Der Browser gibt Folgendes zurück:

```
{
  "id": 4,
  "type": "bird",
  "price": 999.99
}
```

Nächste Schritte

Sie können Ihre API weiter anpassen, indem Sie die Anforderungsvalidierung aktivieren, Daten umwandeln oder benutzerdefinierte Gateway-Antworten erstellen.

In den folgenden Tutorien finden Sie weitere Möglichkeiten zur Anpassung Ihrer API:

- Weitere Informationen zur Anforderungsvalidierung finden Sie unter [Grundlegende Anforderungsvalidierung in API Gateway einrichten](#).
- Weitere Informationen zur Umwandlung von Anforderungs- und Antwort-Nutzlasten finden Sie unter [Tutorial: Ändern Sie die Integrationsanfrage und die Antwort für Integrationen zu AWS service](#).
- Informationen zum Erstellen benutzerdefinierter Gateway-Antworten finden Sie unter [Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole](#).

Tutorial: Erstellen einer REST-API mit einer privaten Integration

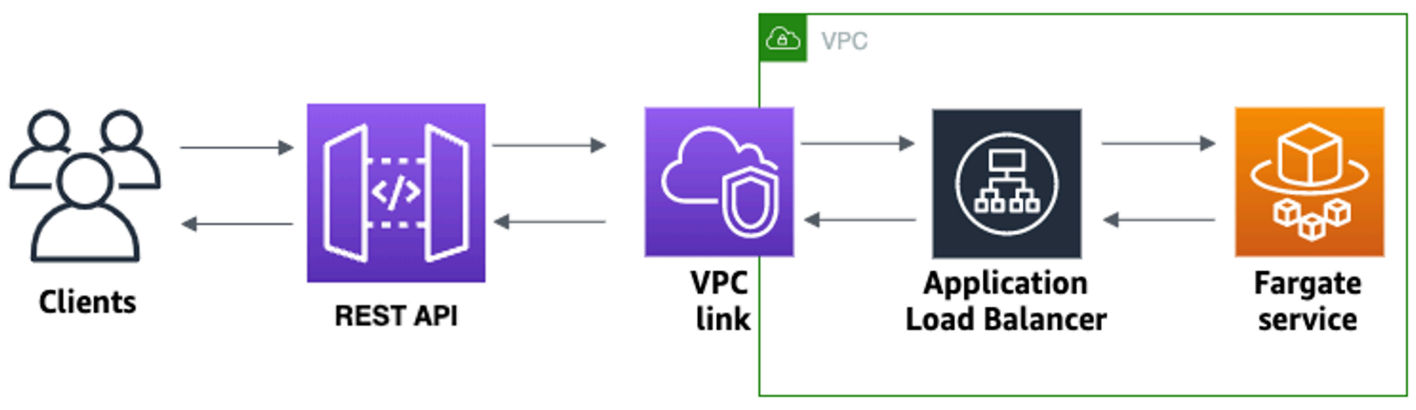
In diesem Tutorial erstellen Sie eine REST-API, die eine Verbindung zu einem Amazon ECS-Service herstellt, der in einer Amazon VPC ausgeführt wird. Clients außerhalb Ihrer Amazon VPC können die API verwenden, um auf Ihren Amazon-ECS-Service zuzugreifen.

Für dieses Tutorial brauchen Sie ungefähr eine Stunde. Zunächst verwenden Sie eine CloudFormation Vorlage, um einen Amazon VPC- und Amazon ECS-Service zu erstellen. Anschließend verwenden Sie die API Gateway Gateway-Konsole, um einen VPC-Link V2 zu erstellen. Der VPC-Link ermöglicht es API Gateway, auf den Amazon-ECS-Service zuzugreifen, der in Ihrer Amazon VPC ausgeführt wird. Als Nächstes erstellen Sie eine REST-API, die den VPC-Link V2 verwendet, um eine Verbindung zu Ihrem Amazon ECS-Service herzustellen. Zuletzt testen Sie Ihre API.

Wenn Sie Ihre REST-API aufrufen, leitet API Gateway die Anfrage über Ihren VPC-Link V2 an Ihren Amazon ECS-Service weiter und gibt dann die Antwort vom Service zurück.

Note

Dieses Tutorial wurde zuvor für HTTP APIs unterstützt und wird jetzt für REST APIs mithilfe von VPC Link V2 unterstützt.



Um dieses Tutorial abzuschließen, benötigen Sie ein AWS Konto und einen AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

Topics

- [Schritt 1: Erstellen Sie einen Amazon-ECS-Service](#)

- [Schritt 2: Erstellen eines VPC-Links](#)
- [Schritt 3: Erstellen Sie eine REST-API](#)
- [Schritt 4: Testen Sie Ihre API](#)
- [Schritt 5: Bereitstellen Ihrer API](#)
- [Schritt 6: Rufen Sie Ihre API auf](#)
- [Schritt 7: Bereinigen](#)

Schritt 1: Erstellen Sie einen Amazon-ECS-Service

Amazon ECS ist ein Container-Management-Service, mit dem Sie unkompliziert Docker-Container in einem Cluster ausführen, beenden und verwalten können. In diesem Tutorial führen Sie Ihren Cluster auf einer serverlosen Infrastruktur aus, die von Amazon ECS verwaltet wird.

Laden Sie [diese CloudFormation Vorlage](#) herunter und entpacken Sie sie, wodurch alle Abhängigkeiten für den Service, einschließlich einer Amazon VPC, erstellt werden. Sie verwenden die Vorlage, um einen Amazon-ECS-Service zu erstellen, der einen Application Load Balancer verwendet.

Um einen Stack zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **rest-api-private-integrations-tutorial** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt den ECS-Service bereit, was einige Minuten dauern kann. Wenn der Status Ihres CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen eines VPC-Links

Ein VPC-Link ermöglicht es API Gateway, auf private Ressourcen in einer Amazon VPC zuzugreifen. Sie verwenden einen VPC-Link, um Kunden den Zugriff auf Ihren Amazon ECS-Service über Ihre REST-API zu ermöglichen.

So erstellen Sie einen VPC-Link

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.

2. Klicken Sie im Hauptnavigationsbereich auf VPC-Links und dann auf Erstellen.

Sie müssen möglicherweise auf das Menüsymbol klicken, um den Hauptnavigationsbereich zu öffnen.

3. Wählen Sie für Wählen Sie eine VPC-Link-Version die Option VPC-Link V2 aus.
4. Geben Sie als Name die Zeichenfolge „**private-integrations-tutorial**“ ein.
5. Wählen Sie für VPC die in Schritt 1 erstellte VPC aus. Der Name muss mit RestApiStack beginnen.
6. Wählen Sie für Subnetze die beiden privaten Subnetze in Ihrer VPC aus. Ihre Namen enden mit PrivateSubnet.
7. Wählen Sie unter Sicherheitsgruppen die Gruppen-ID aus, die mit `private-integrations-tutorial` beginnt und die `RestApiStack/RestApiTutorialService/Service/SecurityGroup`-Beschreibung enthält.
8. Wählen Sie Erstellen aus.

Nachdem Sie Ihren VPC-Link V2 erstellt haben, stellt API Gateway Elastic Network Interfaces für den Zugriff auf Ihre VPC bereit. Der Vorgang kann einige Minuten dauern. In der Zwischenzeit können Sie Ihre API erstellen.

Schritt 3: Erstellen Sie eine REST-API

Die REST-API bietet einen HTTP-Endpunkt für Ihren Amazon ECS-Service.

So erstellen Sie eine REST-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.

2. Wählen Sie API erstellen und dann für Rest-API die Option Erstellen aus.
3. Geben Sie unter Name **private-integration-api** ein.
4. Wählen Sie als IP-Adresstyp die Option. IPv4
5. Wählen Sie Create API (API erstellen) aus.

Nachdem Sie Ihre API erstellt haben, erstellen Sie eine Methode.

6. Wählen Sie Methode erstellen aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie als Methodentyp die Option GET aus.
 - b. Für Integrationstyp wählen Sie VPC-Link aus.
 - c. Aktivieren Sie die Lambda-Proxy-Integration.
 - d. Für HTTP-Methode wählen Sie GET aus.
 - e. Wählen Sie für VPC-Link den VPC-Link V2 aus, den Sie im vorherigen Schritt erstellt haben.
 - f. Geben Sie als Integrationsziel den Load Balancer ein, den Sie mit der CloudFormation Vorlage in Schritt 1 erstellt haben. Sein Name sollte mit rest- beginnen.
 - g. Geben Sie für Endpunkt-URL `http://private-integrations-tutorial.com` ein.

Die URL wird verwendet, um den Host Header der Integrationsanfrage festzulegen. In diesem Fall ist der Host-Header **private-integrations-tutorial**.

- h. Wählen Sie Methode erstellen aus.

Mit der Proxy-Integration ist die API bereit zum Testen.

Schritt 4: Testen Sie Ihre API

Als Nächstes testen Sie den Aufruf der API-Methode.

So testen Sie Ihre API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
4. Wählen Sie Test

Stellen Sie sicher, dass die Antwort Ihrer API eine Willkommensnachricht ist, die Ihnen mitteilt, dass Ihre App auf Amazon ECS ausgeführt wird.

Schritt 5: Bereitstellen Ihrer API

Als Nächstes stellen Sie Ihre API bereit.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **Prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

Schritt 6: Rufen Sie Ihre API auf

Nachdem Ihre API bereitgestellt wurde, können Sie sie aufrufen.

Um Ihre API aufzurufen

1. Geben Sie die Aufruf-URL in einen Webbrowser ein.

Die URL sollte wie folgt aussehen: `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Ihr Browser sendet eine GET-Anforderung an die API.

2. Stellen Sie sicher, dass die Antwort Ihrer API eine Willkommensnachricht ist, die Ihnen mitteilt, dass Ihre App auf Amazon ECS ausgeführt wird.

Wenn Sie die Willkommensnachricht sehen, haben Sie erfolgreich einen Amazon ECS-Service erstellt, der in einer Amazon VPC ausgeführt wird, und Sie haben eine API-Gateway-REST-API mit einem VPC-Link V2 verwendet, um auf den Amazon ECS-Service zuzugreifen.

Schritt 7: Bereinigen

Um unnötige Kosten zu verhindern, löschen Sie die Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben. Mit den folgenden Schritten werden Ihr VPC-Link V2, Ihr CloudFormation Stack und Ihre REST-API gelöscht.

So löschen Sie eine REST-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIs-Seite eine API aus. Wählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

So löschen Sie einen VPC-Link

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie VPC-Link.
3. Wählen Sie Ihren VPC-Link aus, wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Um einen Stapel zu löschen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Tutorial: Erstellen Sie eine REST-API mit einer AWS Integration

In [Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration](#) und [Tutorial: Erstellen einer REST-API mit einer Lambda-Nicht-Proxy-Integration](#) wird beschrieben, wie eine API Gateway-API erstellt wird, um die integrierte Lambda-Funktion zu bereitstellen. Darüber hinaus können Sie eine API-Gateway-API erstellen, um andere AWS Dienste wie Amazon SNS, Amazon S3, Amazon Kinesis usw. verfügbar zu machen. AWS Lambda Dies wird durch die AWS-Integration ermöglicht. Die Lambda-Integration oder die Lambda-Proxy-Integration ist ein Sonderfall, bei dem der Aufruf der Lambda-Funktion über die API Gateway-API bereitgestellt wird.

Alle AWS Dienste unterstützen ausschließlich APIs die Bereitstellung ihrer Funktionen. Die Anwendungsprotokolle oder Programmierschnittstellen sind jedoch wahrscheinlich von Service zu Service unterschiedlich. Eine API-Gateway-API mit der AWS Integration hat den Vorteil, dass sie Ihrem Kunden ein konsistentes Anwendungsprotokoll für den Zugriff auf verschiedene AWS Dienste bietet.

In dieser exemplarischen Vorgehensweise erstellen wir eine API zur Bereitstellung von von Amazon SNS. Weitere Beispiele für die Integration einer API mit anderen AWS Diensten finden Sie unter [Amazon API Gateway-Tutorials und -Workshops](#).

Im Gegensatz zur Lambda-Proxy-Integration gibt es keine entsprechende Proxy-Integration für andere AWS -Services. Daher wird eine API-Methode mit einer einzigen AWS Aktion integriert. Für mehr Flexibilität, ähnlich wie bei der Proxy-Integration, können Sie eine Lambda-Proxy-Integration einrichten. Die Lambda-Funktion analysiert und verarbeitet dann Anfragen für andere AWS Aktionen.

API Gateway versucht es nicht erneut, wenn der Endpunkt das Zeitlimit überschreitet. Der API-Aufrufer muss eine Logik für Wiederholversuche implementieren, um mit Endpunkt-Timeouts umzugehen.

Diese schrittweise Anleitung basiert auf den Anweisungen und Konzepten in [Tutorial: Erstellen einer REST-API mit einer Lambda-Nicht-Proxy-Integration](#). Wenn Sie diese Anleitung noch nicht ausgeführt haben, sollen Sie dies zuerst nachholen.

Themen

- [Voraussetzungen](#)
- [Schritt 1: Erstellen Sie die AWS Service-Proxy-Ausführungsrolle](#)
- [Schritt 2: Erstellen der Ressource](#)
- [Schritt 3: Erstellen der GET-Methode](#)
- [Schritt 4: Angeben von Methodeneinstellungen und Testen der Methode](#)
- [Schritt 5: Bereitstellen der API](#)
- [Schritt 6: Testen der API](#)
- [Schritt 7: Bereinigen](#)

Voraussetzungen

Führen Sie die folgenden Schritte aus, bevor Sie mit der schrittweisen Anleitung beginnen:

1. Führen Sie die Schritte unter [Einrichten der API Gateway](#).
2. Erstellen Sie eine neue API mit dem Namen MyDemoAPI. Weitere Informationen finden Sie unter [Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen](#).
3. Stellen Sie die API mindestens einmal für eine Stufe mit dem Namen bereit test. Weitere Informationen finden Sie unter [Bereitstellen der API](#) in [Wählen Sie ein AWS Lambda Integrations-Tutorial](#).
4. Führen Sie die restlichen Schritte in [Wählen Sie ein AWS Lambda Integrations-Tutorial](#).
5. Erstellen Sie mindestens ein Thema in Amazon Simple Notification Service (Amazon SNS). Sie verwenden die bereitgestellte API, um eine Liste der Themen in Amazon SNS abzurufen, die mit Ihrem AWS Konto verknüpft sind. Wie Sie ein Thema in Amazon SNS erstellen, erfahren Sie unter [Thema erstellen](#). (Sie müssen nicht die in Schritt 5 erwähnte Thema-ARN kopieren.)

Schritt 1: Erstellen Sie die AWS Service-Proxy-Ausführungsrolle

Einer IAM-Rolle müssen geeignete IAM-Richtlinien zugeordnet sein, damit die API Amazon-SNS-Aktionen aufrufen kann. In dieser Übung erstellen Sie eine neue IAM-Rolle.

Um die AWS Service-Proxy-Ausführungsrolle zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie Roles.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS Dienst aus, wählen Sie dann API Gateway aus und wählen Sie Erlaubt API Gateway, Logs in Logs zu CloudWatch übertragen.
5. Klicken Sie auf Weiter und dann erneut auf Weiter.
6. Geben Sie für Role name (Rollenname) den Namen **APIGatewaySNSProxyPolicy** ein und klicken Sie auf Create role (Rolle erstellen).
7. Wählen Sie in der Liste Roles die Rolle aus, die Sie gerade erstellt haben. Möglicherweise müssen Sie scrollen oder die Rolle über die Suchleiste finden.
8. Wählen Sie für die ausgewählte Rolle die Registerkarte Berechtigungen hinzufügen aus.
9. Wählen Sie in der Dropdown-Liste Berechtigungen anfügen aus.

10. Geben Sie im Suchfeld **AmazonSNSReadOnlyAccess** ein und wählen Sie Berechtigungen hinzufügen aus.

 Note

Dieses Tutorial verwendet der Einfachheit halber eine verwaltete Richtlinie. Als Best Practice sollten Sie Ihre eigene IAM-Richtlinie erstellen, um die erforderlichen Mindestberechtigungen zu gewähren.

11. Notieren Sie sich den neu erstellten Rollen-ARN, Sie werden ihn später brauchen.

Schritt 2: Erstellen der Ressource

In diesem Schritt erstellen Sie eine Ressource, die es dem AWS Service-Proxy ermöglicht, mit dem AWS Dienst zu interagieren.

So erstellen Sie die Ressource

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie die Stammressource / aus. Sie erkennen sie an dem einzelnen Schrägstrich (/), und klicken Sie dann auf Ressource erstellen.
4. Die Proxy-Ressource bleibt ausgeschaltet.
5. Ressourcenpfad wird als / beibehalten.
6. Geben Sie für Resource name (Ressourcenname) **mydemoawsproxy** ein.
7. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
8. Wählen Sie Create Resource (Ressource erstellen) aus.

Schritt 3: Erstellen der GET-Methode

In diesem Schritt erstellen Sie eine GET-Methode, die es dem AWS Dienstproxy ermöglicht, mit dem Dienst zu interagieren. AWS

So erstellen Sie die **GET**-Methode

1. Wählen Sie die /mydemoawsproxy-Ressource aus und klicken Sie dann auf Methode erstellen.

2. Wählen Sie als Methodentyp GET aus.
3. Für Integrationstyp wählen Sie AWS-Service aus.
4. Wählen Sie für den Ort aus AWS-Region, AWS-Region an dem Sie Ihr Amazon SNS SNS-Thema erstellt haben.
5. Wählen Sie für AWS-Service Amazon SNS aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Für HTTP-Methode wählen Sie GET aus.
8. Wählen Sie für Aktionstyp die Option Aktionsnamen verwenden aus.
9. Für Aktionsname geben Sie **ListTopics** ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **APIGatewaySNSProxyPolicy** ein.
11. Wählen Sie Methode erstellen aus.

Schritt 4: Angeben von Methodeneinstellungen und Testen der Methode

Sie können nun Ihre GET-Methode testen, um sicherzustellen, dass sie korrekt für die Auflistung Ihrer SNS-Themen eingerichtet wurde.

So testen Sie die **GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie Test aus.

Im Ergebnis wird eine Antwort ähnlich der Folgenden angezeigt:

```
{
  "ListTopicsResponse": {
    "ListTopicsResult": {
      "NextToken": null,
      "Topics": [
        {
          "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-1"
        },
        {
          "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-2"
        },
        ...
      ]
    }
  }
}
```

```
{
  "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-N"
},
"ResponseMetadata": {
  "RequestId": "abc1de23-45fa-6789-b0c1-d2e345fa6b78"
}
}
```

Schritt 5: Bereitstellen der API

In diesem Schritt stellen Sie die API bereit, damit Sie sie von außerhalb der API Gateway-Konsole aufrufen können.

So stellen Sie die API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **test** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Deploy (Bereitstellen) aus.

Schritt 6: Testen der API

In diesem Schritt verlassen Sie die API Gateway Gateway-Konsole und verwenden Ihren AWS Service-Proxy, um mit dem Amazon SNS SNS-Service zu interagieren.

1. Klicken Sie im Hauptnavigationsbereich auf Stufe.
2. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Das sollte wie folgt aussehen:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test
```

3. Geben Sie die URL in das Adressfeld einer neuen Registerkarte im Browser ein.
4. Fügen Sie /mydemoawsproxy an, sodass die URL wie folgt aussieht:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test/mydemoawsproxy
```

Rufen Sie die URL auf. Informationen wie die folgenden werden angezeigt:

```
{"ListTopicsResponse":{"ListTopicsResult":{"NextToken": null,"Topics":
[{"TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-1"}, {"TopicArn":
"arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-2"}, ... {"TopicArn":
"arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-N"}]}, "ResponseMetadata":
{"RequestId": "abc1de23-45fa-6789-b0c1-d2e345fa6b78"}}}
```

Schritt 7: Bereinigen

Sie können die IAM-Ressourcen löschen, die der AWS Service-Proxy benötigt, um zu funktionieren.

Warning

Wenn Sie eine IAM-Ressource löschen, auf die sich ein AWS Service-Proxy stützt, funktionieren dieser AWS Dienstproxy und alle APIs, die darauf angewiesen sind, nicht mehr. Das Löschen einer IAM-Ressource kann nicht rückgängig gemacht werden. Wenn Sie die IAM-Ressource wieder nutzen möchten, müssen Sie diese neu erstellen.

Löschen Sie die zugehörigen IAM-Ressourcen wie folgt:

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Bereich Details die Option Roles aus.
3. Wählen Sie APIGatewayAWSProxyExecRole aus und anschließend Role Actions und Delete Role. Wählen Sie bei Aufforderung Yes, Delete.
4. Wählen Sie im Bereich Details Policies.
5. Wählen Sie APIGatewayAWSProxyExecPolicy und dann anschließend Policy Actions Delete. Wählen Sie bei Aufforderung Löschen.

Sie haben das Ende dieser Anleitung erreicht. Ausführlichere Erläuterungen zum Erstellen einer API als AWS Dienstproxy finden Sie unter, [Tutorial: REST-API als AmazonS3-Proxy erstellen](#), [Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration](#), oder [Tutorial: REST-API als Amazon Kinesis-Proxy erstellen](#).

Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration

Das [the section called “Tutorial: Erstellen einer REST-API mit einer Lambda-Nicht-Proxy-Integration”](#) verwendet ausschließlich Lambda Function-Integration. Die Lambda Function-Integration ist ein Sonderfall des AWS Service-Integrationstyps, der einen Großteil der Integrationseinrichtung für Sie durchführt, z. B. das automatische Hinzufügen der erforderlichen ressourcenbasierten Berechtigungen zum Aufrufen der Lambda-Funktion. Hier wird bei zwei der drei Integrationen die AWS Service-Integration verwendet. Bei diesem Integrationstyp haben Sie mehr Kontrolle, aber Sie müssen bestimmte Aufgaben manuell ausführen. Hierzu gehört z. B. das Erstellen und Angeben einer IAM-Rolle mit entsprechenden Berechtigungen.

In diesem Tutorial erstellen Sie eine Calc-Lambda-Funktion, die grundlegende Rechenoperationen implementiert und dabei Ein- und Ausgaben im JSON-Format annimmt und zurückgibt. Dann erstellen Sie eine REST-API und integrieren sie mit der Lambda-Funktion auf folgende Weise:

1. Verwenden einer GET-Methode für die `/calc`-Ressource, um die Lambda-Funktion aufzurufen, wobei die Eingabe als Abfragezeichenfolge-Parameter übergeben wird. (AWS Service-Integration)
2. Verwenden einer POST-Methode für die `/calc`-Ressource, um die Lambda-Funktion aufzurufen, wobei die Eingabe in der Methodenanforderungsnutzlast übergeben wird. (AWS Service-Integration)
3. Verwenden einer GET-Methode für verschachtelte `/calc/{operand1}/{operand2}/{operator}`-Ressourcen, um die Lambda-Funktion aufzurufen, wobei die Eingabe als Pfadparameter übergeben wird. (Lambda Function-Integration)

Über dieses Tutorial hinaus steht die [OpenAPI-Definitionsdatei](#) für die Calc-API, die Sie in API Gateway importieren können, bereit. Befolgen Sie dazu die Anweisungen in [the section called “OpenAPI”](#).

Themen

- [Übernehmbare IAM-Rolle erstellen](#)
- [Erstellen einer Calc-Lambda-Funktion](#)
- [Calc Lambda-Funktion testen](#)
- [Erstellen einer Calc-API](#)

- [Integration 1: Erstellen Sie eine GET-Methode mit Abfrageparametern, um die Lambda-Funktion aufzurufen](#)
- [Integration 2: Erstellen Sie eine POST-Methode mit einem JSON-Payload für den Aufruf der Lambda-Funktion](#)
- [Integration 3: Erstellen Sie eine GET-Methode mit Pfadparametern für den Aufruf der Lambda-Funktion](#)
- [OpenAPI-Definitionen der mit einer Lambda-Funktion integrierten Beispiel-API](#)

Übernehmbare IAM-Rolle erstellen

Damit Ihre API Ihre Calc Lambda-Funktion aufrufen kann, benötigen Sie eine IAM-Rolle, die von API Gateway übernommen werden kann, d. h. eine IAM-Rolle mit der folgenden vertrauenswürdigen Beziehung:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Die Rolle, die Sie erstellen, benötigt eine [InvokeFunction](#) Lambda-Berechtigung. Andernfalls erhält der API-Aufrufer eine 500 Internal Server Error-Antwort. Um der Rolle diese Berechtigung zu erteilen, hängen ihr die folgende IAM-Richtlinie an:

JSON

```
{
```



```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "lambda:InvokeFunction",
    "Resource": "*"
  }
]
```

Gehen Sie hierzu folgendermaßen vor:

API Gateway erstellen, das die IAM-Rolle übernimmt

1. Melden Sie sich bei der IAM-Konsole an.
2. Wählen Sie Roles.
3. Wählen Sie Create Role aus.
4. Wählen Sie unter Select type of trusted entity (Typ der vertrauenswürdigen Entität auswählen) die Option AWS Service aus.
5. Wählen Sie unter Choose the service that will use this role (Die Rolle auswählen, die diese Rollen verwenden wird) die Option Lambda aus.
6. Wählen Sie Next: Permissions aus.
7. Wählen Sie Create Policy (Richtlinie erstellen) aus.

Ein neues Create Policy Konsolen-Fenster wird geöffnet. Führen Sie in diesem Fenster folgende Schritte aus:

- a. Ersetzen Sie auf der Registerkarte JSON die vorhandene Richtlinie durch Folgendes:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
```

```

        "Resource": "*"
      }
    ]
  }

```

- b. Wählen Sie Review policy (Richtlinie prüfen) aus.
 - c. Gehen Sie unter Review Policy wie folgt vor:
 - i. Geben Sie unter Name einen Namen wie **lambda_execute** ein.
 - ii. Wählen Sie Create Policy (Richtlinie erstellen) aus.
8. Führen Sie im ursprünglichen Konsolen-Fenster Create Role Folgendes aus:
 - a. Wählen Sie unter Attach permissions policies (Berechtigungsrichtlinien hinzufügen), wählen die **lambda_execute** Richtlinie aus der Dropdown-Liste aus.

 Wenn Ihre Richtlinie nicht in der Liste angezeigt wird, wählen Sie die Schaltfläche "Aktualisieren" oben in der Liste aus. (Aktualisieren Sie nicht die Browserseite!)
 - b. Wählen Sie Next: Tags (Weiter: Tags) aus.
 - c. Klicken Sie auf Weiter: Prüfen.
 - d. Geben Sie unter Role name einen Namen wie **lambda_invoke_function_assume_apigw_role** ein.
 - e. Wählen Sie Create role aus.
9. Wählen Sie Ihre **lambda_invoke_function_assume_apigw_role** aus der Rollen-Liste aus.
10. Wählen Sie die Registerkarte Trust relationships (Vertrauensstellungen).
11. Wählen Sie Edit Trust Relationship (Vertrauensstellungen bearbeiten).
12. Ersetzen Sie die vorhandene Richtlinie durch Folgendes:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",

```

```
    "Principal": {
      "Service": [
        "lambda.amazonaws.com",
        "apigateway.amazonaws.com"
      ]
    },
    "Action": "sts:AssumeRole"
  }
]
```

13. Wählen Sie Update Trust Policy.
14. Notieren Sie sich die Rolle ARN für die Rolle, die Sie gerade erstellt haben. Sie benötigen sie zu einem späteren Zeitpunkt.

Erstellen einer **Calc**-Lambda-Funktion

Als Nächstes erstellen Sie eine Lambda-Funktion unter Verwendung der Lambda-Konsole.

1. Wählen Sie in der Lambda-Konsole Create function (Funktion erstellen).
2. Wählen Sie Author from scratch aus.
3. Geben Sie als Name die Zeichenfolge „**Calc**“ ein.
4. Wählen Sie für Laufzeit die neueste unterstützte Node.js- oder Python-Laufzeit aus.
5. Verwenden Sie die Standardeinstellung für alle anderen Optionen.
6. Wählen Sie Funktion erstellen.
7. Kopieren Sie die folgende Lambda-Funktion in Ihre bevorzugte Laufzeit und fügen Sie sie in den Code-Editor in der Lambda-Konsole ein.

Node.js

```
export const handler = async function (event, context) {
  console.log("Received event:", JSON.stringify(event));

  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
```

```
    return "400 Invalid Input";
  }

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
    case "add":
      res.c = res.a + res.b;
      break;
    case "-":
    case "sub":
      res.c = res.a - res.b;
      break;
    case "*":
    case "mul":
      res.c = res.a * res.b;
      break;
    case "/":
    case "div":
      if (res.b == 0) {
        return "400 Divide by Zero";
      } else {
        res.c = res.a / res.b;
      }
      break;
    default:
      return "400 Invalid Operator";
  }

  return res;
};
```

Python

```
import json
```

```
def lambda_handler(event, context):
    print(event)

    try:
        (event['a']) and (event['b']) and (event['op'])
    except KeyError:
        return '400 Invalid Input'

    try:
        res = {
            "a": float(
                event['a']), "b": float(
                event['b']), "op": event['op']}
    except ValueError:
        return '400 Invalid Operand'

    if event['op'] == '+':
        res['c'] = res['a'] + res['b']
    elif event['op'] == '-':
        res['c'] = res['a'] - res['b']
    elif event['op'] == '*':
        res['c'] = res['a'] * res['b']
    elif event['op'] == '/':
        if res['b'] == 0:
            return '400 Divide by Zero'
        else:
            res['c'] = res['a'] / res['b']
    else:
        return '400 Invalid Operator'

    return res
```

8. Wählen Sie unter "Ausführungsrolle" Choose an existing role (Wählen Sie eine vorhandene Rolle aus) aus.
9. Geben Sie den Rollen-ARN für die zuvor erstellte **lambda_invoke_function_assume_apigw_role**-Rolle ein.
10. Wählen Sie Deploy (Bereitstellen) aus.

Diese Funktion erfordert zwei Operanden (a und b) und einen Operator (op) aus dem Eingabeparameter event. Die Eingabe ist ein JSON-Objekt im folgenden Format:

```
{
  "a": "Number" | "String",
  "b": "Number" | "String",
  "op": "String"
}
```

Diese Funktion gibt das berechnete Ergebnis (c) und die Eingabe zurück. Im Falle einer ungültigen Eingabe gibt die Funktion entweder den Wert null (0) oder die Zeichenfolge "Invalid op" als Ergebnis zurück. Die Ausgabe erfolgt im folgenden JSON-Format:

```
{
  "a": "Number",
  "b": "Number",
  "op": "String",
  "c": "Number" | "String"
}
```

Testen Sie die Funktion in der Lambda-Konsole, bevor Sie diese im nächsten Schritt in die API integrieren.

Calc Lambda-Funktion testen

Hier erfahren Sie, wie Sie Ihre Calc Funktion in der Lambda-Konsole testen können:

1. Wählen Sie die Registerkarte Test.
2. Geben Sie für den Testereignisnamen ei **calc2plus5**.
3. Ersetzen Sie die Testereignisdefinition durch Folgendes:

```
{
  "a": "2",
  "b": "5",
  "op": "+"
}
```

4. Wählen Sie Save (Speichern) aus.
5. Wählen Sie Test aus.

6. Erweitern Sie Execution result: succeeded (Ausführungsergebnis: erfolgreich). Sie sollten Folgendes sehen:

```
{
  "a": 2,
  "b": 5,
  "op": "+",
  "c": 7
}
```

Erstellen einer **Calc**-API

Das folgende Verfahren zeigt, wie Sie eine API für die gerade erstellte Calc Lambda-Funktion erstellen. In den folgenden Abschnitten fügen Sie dieser Ressourcen und Methoden hinzu.

So erstellen Sie eine API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Geben Sie in API name (API-Name) **LambdaCalc** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunktyp bei Regional.
6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Integration 1: Erstellen Sie eine **GET**-Methode mit Abfrageparametern, um die Lambda-Funktion aufzurufen

Durch Erstellen einer GET-Methode, die Abfragezeichenfolgenparameter an die Lambda-Funktion übergibt, wird das Aufrufen der API über einen Browser aktiviert. Dieser Ansatz kann nützlich sein, insbesondere für solche APIs, die offenen Zugriff ermöglichen.

Nachdem Sie eine API erstellt haben, können Sie nun eine Ressource erstellen. In der Regel werden API-Ressourcen in einer Ressourcenstruktur organisiert, die der Anwendungslogik entspricht. In diesem Schritt erstellen Sie eine /calc-Ressource.

So erstellen Sie eine /calc-Ressource

1. Wählen Sie Create Resource (Ressource erstellen) aus.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Ressourcenpfad wird als / beibehalten.
4. Geben Sie für Resource name (Ressourcenname) **calc** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

Durch Erstellen einer GET-Methode, die Abfragezeichenfolgenparameter an die Lambda-Funktion übergibt, wird das Aufrufen der API über einen Browser aktiviert. Dieser Ansatz kann sich als nützlich erweisen, insbesondere für solche APIs, die einen offenen Zugang ermöglichen.

Bei dieser Methode muss die POST-Anforderung verwendet werden, um eine Lambda-Funktion aufzurufen. Das Beispiel verdeutlicht, dass die HTTP-Methoden einer Frontend-Methodenanforderung und einer Backend-Integrationsanforderung unterschiedlich sein können.

So erstellen Sie eine **GET**-Methode

1. Wählen Sie die /calc-Ressource und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Für Integrationstyp wählen Sie AWS-Service aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihre Lambda-Funktion erstellt haben.
5. Wählen Sie für AWS-Service Lambda aus.

6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie für HTTP-Methode POST aus.
8. Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus. Mit dieser Option wird der ARN der [Invoke](#)-Aktion für die Ausführung der Calc-Funktion angegeben.
9. Geben Sie für Pfadüberschreibung **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations** ein. Geben Sie für **account-id** Ihre AWS-Konto ID ein. Geben Sie für den Ort ein **us-east-2**, AWS-Region an dem Sie Ihre Lambda-Funktion erstellt haben.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **lambda_invoke_function_assume_apigw_role** ein.
11. Die Einstellungen für Credential-Cache und Standard-Timeout bleiben unverändert.
12. Klicken Sie auf Einstellungen für Methodenanfragen.
13. Wählen Sie für Anforderungs-Validator die Option Abfragezeichenfolgeparameter und -Header validieren aus.

Mit dieser Einstellung wird eine Fehlermeldung zurückgegeben, falls der Client die erforderlichen Parameter nicht vorgibt.

14. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

Jetzt können Sie Parameter für die Abfragezeichenfolgen der GET-Methode in der /calc-Ressource einrichten sodass diese Eingaben im Namen der Backend-Lambda-Funktion empfangen kann.

Gehen Sie wie folgt vor, um die Parameter für die Abfragezeichenfolge zu erstellen:

- a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
- b. Geben Sie unter Name **operand1** ein.
- c. Aktivieren Sie die Option Erforderlich.
- d. Caching bleibt ausgeschaltet.

Wiederholen Sie die obigen Schritte und erstellen Sie eine Abfragezeichenfolge mit dem Namen **operand2** sowie eine Abfragezeichenfolge mit dem Namen **operator**.

15. Wählen Sie Methode erstellen aus.

Als Nächstes erstellen Sie eine Zuordnungsvorlage, um die vom Client bereitgestellten Abfragezeichenfolgen wie für die `Calc`-Funktion erforderlich in die Integrationsanforderungs-Nutzlast zu übersetzen. Diese Vorlage weist die drei in Method Request (Methodenanfrage) deklarierten Abfragezeichenfolge-Parameter in entsprechende Eigenschaftswerte des JSON-Objekts um, das als Eingabe für die Lambda-Funktion im Backend dient. Das umgewandelte JSON-Objekt wird als Nutzlast der Integrationsanforderung eingebunden.

Zuordnen von Eingabeparametern zur Integrationsanforderung

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.
3. Wählen Sie Zuordnungsvorlagen aus.
4. Wählen Sie Add mapping template.
5. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
6. Geben Sie für Vorlagentext den folgenden Code ein:

```
{
  "a": "$input.params('operand1')",
  "b": "$input.params('operand2')",
  "op": "$input.params('operator')"
}
```

7. Wählen Sie Speichern.

Sie können nun Ihre GET-Methode testen, um zu verifizieren, dass sie korrekt für einen Lambda-Funktionsaufruf eingerichtet wurde.

So testen Sie die **GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Abfrage-Zeichenketten geben Sie **operand1=2&operand2=3&operator=+** ein.
3. Wählen Sie Test aus.

Die Ergebnisse sollten in etwa wie folgt aussehen:

/calc - GET method test results		
Request	Latency ms	Status
/calc? operand1=2&operand2=3&operator=+	555	200
Response body		
{ "a": 2, "b": 3, "op": "+", "c": 5 }		

Integration 2: Erstellen Sie eine **POST**-Methode mit einem JSON-Payload für den Aufruf der Lambda-Funktion

Wenn Sie eine POST-Methode mit einer JSON-Nutzlast für den Lambda-Funktionsaufruf erstellen, gehen Sie so vor, dass der Client die erforderlichen Eingaben im Anfragetext an die Backend-Funktion übermitteln muss. Damit sichergestellt ist, dass der Client die korrekten Eingabedaten hochlädt, aktivieren Sie die Anforderungvalidierung für die Nutzlast.

Erstellen einer **POST** -Methode mit einer JSON-Nutzlast

1. Wählen Sie die /calc-Ressource und klicken Sie dann auf Methode erstellen.
2. Wählen Sie unter Method type (Methodentyp) die Option POST aus.
3. Für Integrationstyp wählen Sie AWS-Service aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihre Lambda-Funktion erstellt haben.
5. Wählen Sie für AWS-Service Lambda aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie für HTTP-Methode POST aus.
8. Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus. Mit dieser Option wird der ARN der [Invoke](#)-Aktion für die Ausführung der Calc-Funktion angegeben.
9. Geben Sie für Pfadüberschreibung **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations** ein. Geben Sie für **account-id** Ihre AWS-Konto ID ein. Geben Sie für den Ort ein **us-east-2**, AWS-Region an dem Sie Ihre Lambda-Funktion erstellt haben.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **lambda_invoke_function_assume_apigw_role** ein.
11. Die Einstellungen für Credential-Cache und Standard-Timeout bleiben unverändert.

12. Wählen Sie Methode erstellen aus.

Sie können nun ein Eingabe-Modell erstellen, das die Struktur der Eingabedaten beschreibt und den eingehenden Anfragetext validiert.

Erstellen eines Eingabemodells

1. Klicken Sie im Navigationsbereich auf Models (Modelle).
2. Wählen Sie Modell erstellen aus.
3. Geben Sie unter Name **input** ein.
4. Geben Sie für Content type (Inhaltstyp) **application/json** ein.

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

5. Geben Sie für Modellschema Folgendes ein:

```
{
  "type": "object",
  "properties": {
    "a": { "type": "number" },
    "b": { "type": "number" },
    "op": { "type": "string" }
  },
  "title": "input"
}
```

6. Wählen Sie Modell erstellen aus.

Als Nächstes erstellen Sie ein Ausgabe-Modell. Dieses Modell definiert die Datenstruktur der berechneten Ausgabe vom Backend. Damit werden die Integrationsantwortdaten einem anderen Modell zugewiesen. In diesem Tutorial wird das Pass-Through-Verhalten verwendet, daher wird dieses Modell nicht benötigt.

Erstellen eines Ausgabemodells

1. Wählen Sie Modell erstellen aus.
2. Geben Sie unter Name **output** ein.

3. Geben Sie für Content type (Inhaltstyp) **application/json** ein.

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

4. Geben Sie für Modellschema Folgendes ein:

```
{
  "type": "object",
  "properties": {
    "c": { "type": "number" }
  },
  "title": "output"
}
```

5. Wählen Sie Modell erstellen aus.

Als Nächstes erstellen Sie ein Ergebnis-Modell. Dieses Modell definiert die Datenstruktur der zurückgegebenen Antwortdaten. Dieses Modell verweist auf die Eingabe- und Ausgabe-Schemata, die in Ihrer API definiert sind.

Erstellen eines Ergebnismodells

1. Wählen Sie Modell erstellen aus.
2. Geben Sie unter Name **result** ein.
3. Geben Sie für Content type (Inhaltstyp) **application/json** ein.

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

4. Geben Sie für Modellschema das folgende Modell mit Ihrem *restapi-id* ein. Ihr *restapi-id* ist im folgenden Schema oben in der Konsole in Klammern aufgeführt: API Gateway > APIs > LambdaCalc (*abc123*).

```
{
  "type": "object",
  "properties": {
    "input": {
```

```
        "$ref": "https://apigateway.amazonaws.com/restapis/restapi-id/models/  
input"  
    },  
    "output": {  
        "$ref": "https://apigateway.amazonaws.com/restapis/restapi-id/models/  
output"  
    }  
    },  
    "title": "result"  
}
```

5. Wählen Sie Modell erstellen aus.

Als Nächstes konfigurieren Sie die Methodenanforderung Ihrer POST-Methode, um die Anforderungsvalidierung für den eingehenden Anforderungstext zu aktivieren.

Anforderungsvalidierung für die POST-Methode aktivieren

1. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die POST-Methode in der Ressourcenstruktur aus.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Wählen Sie für Anforderungs-Validator die Option Text validieren aus.
4. Wählen Sie Anforderungstext aus und klicken Sie dann auf Modell hinzufügen.
5. Geben Sie für Content type (Inhaltstyp) **application/json** ein.

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

6. Wählen Sie für Modell Eingabe aus.
7. Wählen Sie Speichern.

Sie können nun Ihre POST-Methode testen, um zu verifizieren, dass sie korrekt für einen Lambda-Funktionsaufruf eingerichtet wurde.

So testen Sie die **POST**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Geben Sie für Anforderungstext die folgende JSON-Nutzlast ein:

```
{
  "a": 1,
  "b": 2,
  "op": "+"
}
```

3. Wählen Sie Test aus.

Die Ausgabe sollte folgendermaßen aussehen:

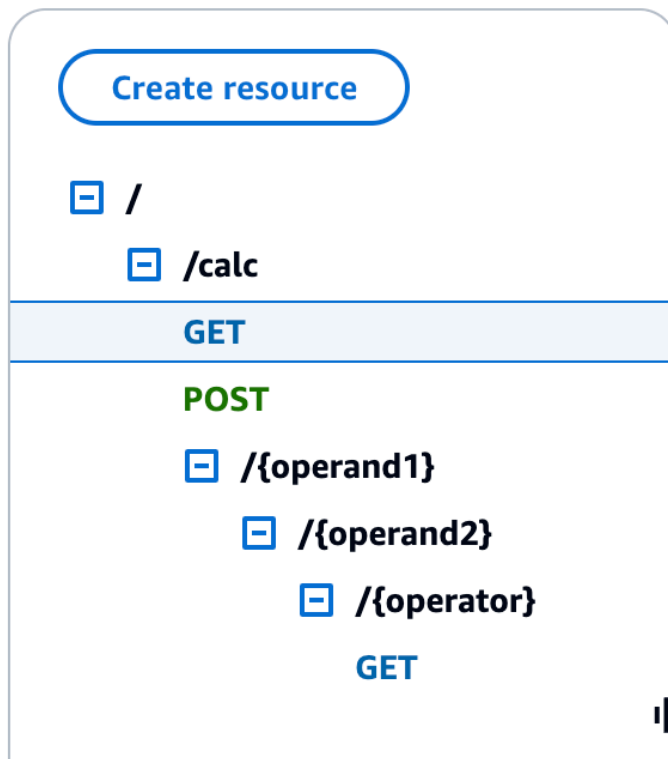
```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```

Integration 3: Erstellen Sie eine **GET**-Methode mit Pfadparametern für den Aufruf der Lambda-Funktion

Jetzt erstellen Sie für eine Ressource eine GET-Methode, die durch eine Abfolge von Pfadparametern angegeben wird, um damit die Lambda-Funktion im Backend aufzurufen. Die Werte der Pfadparameter geben die Eingabedaten für die Lambda-Funktion an. Sie definieren eine Mapping-Vorlage, um die eingehenden Pfadparameterwerte der erforderlichen Integrationsanforderungsnutzlast zuzuweisen.

Die resultierende API-Ressourcen-Struktur sieht etwa wie folgt aus:

Resources



Erstellen einer `/ {operand1} / {operand2} / {operator}`-Ressource

1. Wählen Sie **Create Resource** (Ressource erstellen) aus.
2. Wählen Sie für Ressourcenpfad `/calc` aus.
3. Geben Sie für Resource name (Ressourcenname) **`{operand1}`** ein.
4. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
5. Wählen Sie **Create Resource** (Ressource erstellen) aus.
6. Wählen Sie für Ressourcenpfad `/calc/{operand1}/` aus.
7. Geben Sie für Resource name (Ressourcenname) **`{operand2}`** ein.
8. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
9. Wählen Sie **Create Resource** (Ressource erstellen) aus.
10. Wählen Sie für Ressourcenpfad `/calc/{operand1}/{operand2}/` aus.
11. Geben Sie für Resource name (Ressourcenname) **`{operator}`** ein.
12. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
13. Wählen Sie **Create Resource** (Ressource erstellen) aus.

Dieses Mal verwenden Sie die standardmäßige Lambda-Integration in der API Gateway-Konsole zur Einrichtung der Methodenintegration.

Einrichten einer Integrationsmethode

1. Wählen Sie die `/operand1/operand2/operator`-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Wählen Sie für den Integrationstyp die Option Lambda aus.
4. Lassen Sie Lambda proxy integration (Lambda-Proxyintegration) deaktiviert.
5. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Funktion erstellt haben, und geben Sie ein. **calc**
6. Das Standard-Timeout bleibt aktiviert.
7. Wählen Sie Methode erstellen aus.

Als Nächstes erstellen Sie eine Zuweisungsvorlage zur Zuweisung der drei URL-Pfadparameter, die beim Erstellen der `/calc/operand1/operand2/operator`-Ressource deklariert wurden. Mit dieser Vorlage werden dem JSON-Objekt die vorgegebenen Eigenschaftswerte zugewiesen. Da URL-Pfade eine URL-Codierung erfordern, muss der Operator für die Division mit `%2F` (anstatt `/`) angegeben werden. Diese Vorlage übersetzt den `%2F` in `'/'`, bevor er an die Lambda-Funktion übergeben wird.

Erstellen einer Zuweisungsvorlage

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.
3. Klicken Sie auf Zuordnungsvorlage hinzufügen.
4. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
5. Geben Sie für Vorlagentext den folgenden Code ein:

```
{
  "a": "$input.params('operand1')",
  "b": "$input.params('operand2')",
  "op":
    #if($input.params('operator')=='%2F')"/"#else}"$input.params('operator')"#end
```

```
}
```

6. Wählen Sie Speichern.

Sie können nun Ihre GET-Methode testen, um zu verifizieren, dass sie korrekt für den Aufruf der Lambda-Funktion eingerichtet wurde, und dass die ursprüngliche Ausgabe ohne Zuweisung durch die Integrationsantwort weitergegeben wird.

So testen Sie die **GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Pfad gehen Sie wie folgt vor:
 - a. Geben Sie für operand1 **1** ein.
 - b. Geben Sie für operand2 **1** ein.
 - c. Geben Sie für operator **+** ein.
3. Wählen Sie Test aus.
4. Das Ergebnis sollte wie folgt aussehen:

 <code>/calc/{operand1}/{operand2}/{operator}</code> GET method test results		
Request	Latency ms	Status
<code>/calc/1/1/+</code>	285	200
Response body		
<code>{"a":1,"b":1,"op":"+","c":2}</code>		

Als Nächstes modellieren Sie die Datenstruktur für die Nutzlast der Methodenantwort nach dem `result`-Schema.

Standardmäßig wird dem Methodenantworttext ein leeres Modell zugeordnet. Dadurch wird der Integrationsantworttext ohne Zuweisung übergeben. Wenn Sie jedoch ein SDK für eine stark typisierte Sprache (wie z. B. Java oder Objective-C) erstellen, wird den SDK-Benutzern ein leeres Objekt als Ergebnis übergeben. Damit sichergestellt wird, dass sowohl REST- als auch SDK-Clients das gewünschte Ergebnis erhalten, müssen die Antwortdaten mit einem vordefinierten Schema modelliert werden. Nachfolgend definieren Sie ein Modell für den Methodenantworttext und zum Erstellen einer Mapping-Vorlage, mit der der Integrationsantworttext in den Methodenantworttext umgewandelt wird.

Erstellen einer Methodenantwort

1. Klicken Sie auf der Registerkarte Methodenantwort unter Antwort 200 auf Bearbeiten.
2. Klicken Sie unter Anforderungstext auf Modell hinzufügen.
3. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
4. Wählen Sie für Modell Ergebnis aus.
5. Wählen Sie Speichern.

Durch die Definition eines Modells für den Methodenantworttext wird sichergestellt, dass die Antwortdaten an das `result`-Objekt eines bestimmten SDKs übergeben werden. Um sicherzustellen, dass die Integrationsantwortdaten entsprechend zugewiesen werden, benötigen Sie eine Mapping-Vorlage.

Erstellen einer Zuweisungsvorlage

1. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
2. Klicken Sie auf Zuordnungsvorlage hinzufügen.
3. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
4. Geben Sie für Vorlagentext den folgenden Code ein:

```
#set($inputRoot = $input.path('$'))
{
  "input" : {
    "a" : $inputRoot.a,
    "b" : $inputRoot.b,
    "op" : "$inputRoot.op"
  },
  "output" : {
    "c" : $inputRoot.c
  }
}
```

5. Wählen Sie Speichern.

Testen der Zuweisungsvorlage

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

2. Für Pfad gehen Sie wie folgt vor:
 - a. Geben Sie für operand1 **1** ein.
 - b. Geben Sie für operand2 **2** ein.
 - c. Geben Sie für operator **+** ein.
3. Wählen Sie Test aus.
4. Die Antwort wird wie folgt aussehen:

```
{
  "input": {
    "a": 1,
    "b": 2,
    "op": "+"
  },
  "output": {
    "c": 3
  }
}
```

Momentan kann die API nur über die Test-Funktion in der API Gateway-Konsole aufgerufen werden. Wenn Sie die API für Clients verfügbar machen möchten, muss sie ihnen bereitgestellt werden. Stellen Sie immer sicher, dass Sie Ihre API erneut bereitstellen, wenn Sie eine Ressource oder Methode hinzufügen, ändern oder löschen, oder wenn Sie eine Datenzuweisung oder Stufeneinstellungen aktualisieren. Ohne diesen Schritt werden neue Features oder Aktualisierungen Ihrer API nicht für Clients verfügbar gemacht. Gehen Sie wie folgt vor:

So stellen Sie die API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **Prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.
6. (Optional) Klicken Sie unter Stufendetails auf das Kopiersymbol für Aufruf-URL, um die URL Ihrer Aufruf-API zu kopieren. Sie können diese mit Tools wie [Postman](#) und [cURL](#) zum Testen Ihrer API verwenden.

 Note

Denken Sie daran, dass Sie Ihre API jedes Mal erneut bereitstellen müssen, wenn Sie eine Ressource oder Methode hinzufügen, ändern oder löschen, oder wenn Sie eine Datenzuweisung oder Stufeneinstellungen aktualisieren. Andernfalls sind neue Funktionen oder Aktualisierungen nicht für Clients Ihrer API verfügbar.

OpenAPI-Definitionen der mit einer Lambda-Funktion integrierten Beispiel-API

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-04-20T04:08:08Z",
    "title": "LambdaCalc"
  },
  "host": "uojnr9hd57.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/calc": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "operand2",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operator",
```

```

        "in": "query",
        "required": true,
        "type": "string"
    },
    {
        "name": "operand1",
        "in": "query",
        "required": true,
        "type": "string"
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "schema": {
            "$ref": "#/definitions/Result"
        },
        "headers": {
            "operand_1": {
                "type": "string"
            },
            "operand_2": {
                "type": "string"
            },
            "operator": {
                "type": "string"
            }
        }
    }
},
"x-amazon-apigateway-request-validator": "Validate query string parameters
and headers",
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200",
            "responseParameters": {
                "method.response.header.operator": "integration.response.body.op",
                "method.response.header.operand_2": "integration.response.body.b",
                "method.response.header.operand_1": "integration.response.body.a"
            },
            "responseTemplates": {

```

```

        "application/json": "#set($res = $input.path('$'))\n{\n    \"result\n\": \"$res.a, $res.b, $res.op => $res.c\", \n    \"a\" : \"$res.a\", \n    \"b\" :\n    \"$res.b\", \n    \"op\" : \"$res.op\", \n    \"c\" : \"$res.c\" \n}\n"
    }
}
},
"uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/\narn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
"passthroughBehavior": "when_no_match",
"httpMethod": "POST",
"requestTemplates": {
    "application/json": "{\n    \"a\": \"$input.params('operand1')\", \n\n    \"b\": \"$input.params('operand2')\", \n    \"op\": \"$input.params('operator')\" \n}\n"
},
"type": "aws"
}
},
"post": {
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "parameters": [
        {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
                "$ref": "#/definitions/Input"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "schema": {
                "$ref": "#/definitions/Result"
            }
        }
    }
},
"x-amazon-apigateway-request-validator": "Validate body",

```

```

    "x-amazon-apigateway-integration": {
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "responses": {
        "default": {
          "statusCode": "200",
          "responseTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n  \"a\n\" : $inputRoot.a,\n  \"b\" : $inputRoot.b,\n  \"op\" : $inputRoot.op,\n  \"c\" : $inputRoot.c\n}"
          }
        }
      },
      "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
      "passthroughBehavior": "when_no_templates",
      "httpMethod": "POST",
      "type": "aws"
    }
  },
  "/calc/{operand1}/{operand2}/{operator}": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "operand2",
          "in": "path",
          "required": true,
          "type": "string"
        },
        {
          "name": "operator",
          "in": "path",
          "required": true,
          "type": "string"
        },
        {
          "name": "operand1",
          "in": "path",

```



```

        "required": true,
        "type": "string"
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "schema": {
            "$ref": "#/definitions/Result"
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200",
            "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
\n  \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
\n  \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
            }
        }
    },
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
    "passthroughBehavior": "when_no_templates",
    "httpMethod": "POST",
    "requestTemplates": {
        "application/json": "{\n  \"a\": \"$input.params('operand1')\",\n
\n  \"b\": \"$input.params('operand2')\",\n  \"op\":
\n  #if($input.params('operator')=='%2F')\"/\">{else}\"$input.params('operator')\"#end
\n  }\n}"
    },
    "contentHandling": "CONVERT_TO_TEXT",
    "type": "aws"
  }
}
},
"definitions": {
    "Input": {
        "type": "object",
        "required": [

```

```
        "a",
        "b",
        "op"
    ],
    "properties": {
        "a": {
            "type": "number"
        },
        "b": {
            "type": "number"
        },
        "op": {
            "type": "string",
            "description": "binary op of ['+', 'add', '-', 'sub', '*', 'mul', '%2F',
'div']"
        }
    },
    "title": "Input"
},
"Output": {
    "type": "object",
    "properties": {
        "c": {
            "type": "number"
        }
    },
    "title": "Output"
},
"Result": {
    "type": "object",
    "properties": {
        "input": {
            "$ref": "#/definitions/Input"
        },
        "output": {
            "$ref": "#/definitions/Output"
        }
    },
    "title": "Result"
}
},
"x-amazon-apigateway-request-validators": {
    "Validate body": {
        "validateRequestParameters": false,
```

```
    "validateRequestBody": true
  },
  "Validate query string parameters and headers": {
    "validateRequestParameters": true,
    "validateRequestBody": false
  }
}
```

Tutorial: REST-API als AmazonS3-Proxy erstellen

Dieser Abschnitt zeigt anhand eines Beispiels die Verwendung einer REST-API in API Gateway als Amazon S3-Proxy und beschreibt das Erstellen und Konfigurieren einer REST-API zur Bereitstellung der folgenden Amazon S3-Operationen:

- Bereitstellung von GET bei der Stammressource der API, um [alle Amazon S3-Buckets eines Aufrufers aufzulisten](#).
- Bereitstellung von GET bei einer Folder-Ressource, um [eine Liste aller Objekte in einem Amazon S3-Bucket anzuzeigen](#).
- Machen Sie GET auf einer Folder/Item Ressource verfügbar, um [ein Objekt aus einem Amazon S3 S3-Bucket anzuzeigen oder herunterzuladen](#).

Unter Umständen sollten Sie die Beispiel-API wie in [OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy](#) gezeigt als Amazon-S3-Proxy importieren. Dieses Beispiel enthält weitere exponierte Methoden. Anweisung zum Importieren einer API mithilfe der OpenAPI-Definition finden Sie unter [REST-APIs mit OpenAPI in API Gateway erstellen](#).

Note

Um Ihre API Gateway-API in Amazon S3 zu integrieren, müssen Sie eine Region auswählen, in der sowohl die API Gateway- als auch die Amazon S3-Services verfügbar sind. Informationen zur Verfügbarkeit der Regionen finden Sie unter [Amazon API Gateway-Endpunkte und -Kontingente](#).

Themen

- [Einrichten von IAM-Berechtigungen für die API, um Amazon S3-Aktionen aufzurufen](#)
- [Erstellen von API-Ressourcen zum Darstellen von Amazon S3-Ressourcen](#)
- [Bereitstellen einer API-Methode zur Auflistung der Amazon S3-Buckets des Aufrufers](#)
- [Bereitstellen von API-Methoden, um auf einen Amazon S3-Bucket zuzugreifen](#)
- [Bereitstellen von API-Methoden für den Zugriff auf ein Amazon S3-Objekt in einem Bucket](#)
- [OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy](#)
- [Aufrufen der API mit einem REST-API-Client](#)

Einrichten von IAM-Berechtigungen für die API, um Amazon S3-Aktionen aufzurufen

Einer IAM-Rolle müssen geeignete IAM-Richtlinien zugeordnet sein, damit die API erforderliche Amazon-S3-Aktionen aufrufen kann. In dieser Übung erstellen Sie eine neue IAM-Rolle.

Um die AWS Service-Proxy-Ausführungsrolle zu erstellen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie Roles.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS Dienst aus, wählen Sie dann API Gateway aus und wählen Sie Erlaubt API Gateway, Logs in Logs zu CloudWatch übertragen.
5. Klicken Sie auf Weiter und dann erneut auf Weiter.
6. Geben Sie für Role name (Rollenname) den Namen **APIGatewayS3ProxyPolicy** ein und klicken Sie auf Create role (Rolle erstellen).
7. Wählen Sie in der Liste Roles die Rolle aus, die Sie gerade erstellt haben. Möglicherweise müssen Sie scrollen oder die Rolle über die Suchleiste finden.
8. Wählen Sie für die ausgewählte Rolle die Registerkarte Berechtigungen hinzufügen aus.
9. Wählen Sie in der Dropdown-Liste Berechtigungen anfügen aus.
10. Geben Sie im Suchfeld **AmazonS3FullAccess** ein und wählen Sie Berechtigungen hinzufügen aus.

 Note

Dieses Tutorial verwendet der Einfachheit halber eine verwaltete Richtlinie. Als Best Practice sollten Sie Ihre eigene IAM-Richtlinie erstellen, um die erforderlichen Mindestberechtigungen zu gewähren.

11. Notieren Sie sich den neu erstellten Rollen-ARN, Sie werden ihn später brauchen.

Erstellen von API-Ressourcen zum Darstellen von Amazon S3-Ressourcen

Die Root-Ressource der API (/) verwenden Sie als den Container des Amazon-S3-Buckets eines authentifizierten Aufrufers. Außerdem erstellen Sie die Ressourcen `Folder` und `Item`, um einen bestimmten Amazon-S3-Bucket bzw. ein bestimmtes Amazon-S3-Objekt darzustellen. Der Ordnername und der Objektschlüssel werden vom Aufrufer in Form von Pfadparametern als Teil einer Anforderungs-URL angegeben.

 Note

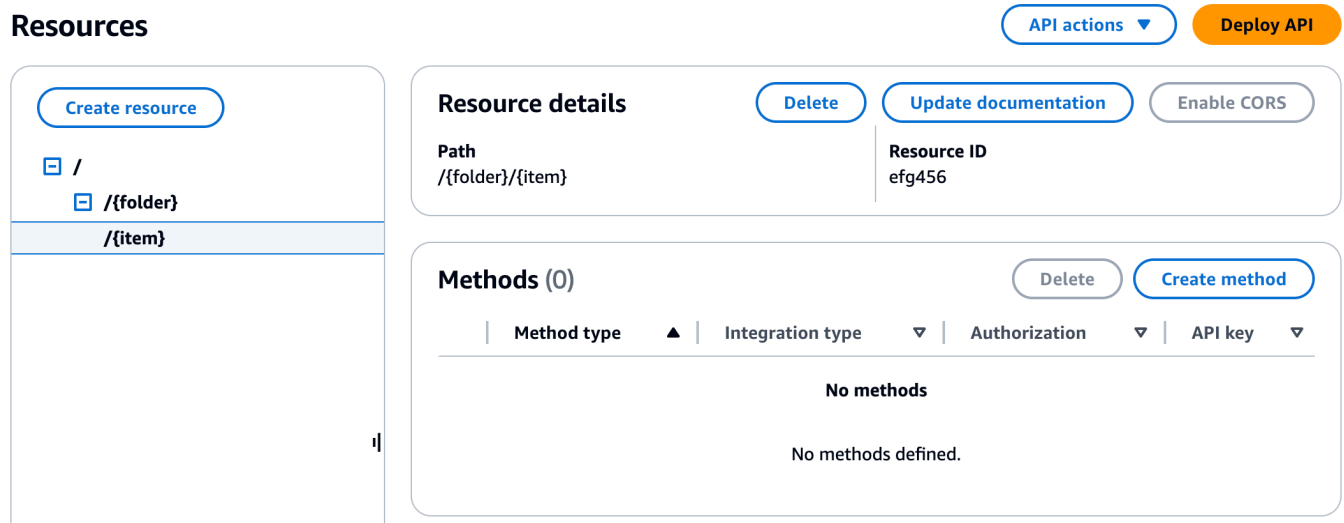
Beim Zugriff auf Objekte, deren Objektschlüssel / oder andere Sonderzeichen enthält, müssen diese Zeichen URL-kodiert sein. Beispielsweise sollte `test/test.txt` zu `test%2Ftest.txt` kodiert werden.

So erstellen Sie eine API-Ressource, die die Amazon S3-Servicefunktionen bereitstellt

1. Erstellen Sie in derselben Weise, in der AWS-Region Sie Ihren Amazon S3 S3-Bucket erstellt haben, eine API mit dem Namen `myS3`. Die Stammressource dieser API (/) stellt den Amazon S3-Service dar. In diesem Schritt erstellen Sie zwei zusätzliche Ressourcen: `/folder` und `/item`.
2. Wählen Sie `Create Resource` (Ressource erstellen) aus.
3. Die `Proxy-Ressource` bleibt ausgeschaltet.
4. Wählen Sie für Ressourcenpfad `/` aus.
5. Geben Sie für Resource name (Ressourcenname) **`folder`** ein.
6. `CORS` (Cross Origin Resource Sharing) bleibt unmarkiert.
7. Wählen Sie `Create Resource` (Ressource erstellen) aus.
8. Wählen Sie die `/folder`-Ressource aus und klicken Sie dann auf `Ressource erstellen`.

- Wiederholen Sie die Schritte oben, um eine untergeordnete Ressource für `/{{folder}}` mit dem Namen `{item}` zu erstellen.

Die endgültige API sollte wie folgt aussehen:



Bereitstellen einer API-Methode zur Auflistung der Amazon S3-Buckets des Aufrufers

Um die Liste der Amazon S3-Buckets des Aufrufers zu erhalten, muss die Aktion [GET Service](#) in Amazon S3 aufgerufen werden. Erstellen Sie die GET-Methode in der Stammressource der API (/). Konfigurieren Sie die GET-Methode für die Integration in Amazon S3 wie folgt.

So erstellen und initialisieren Sie die **GET /**-Methode der API

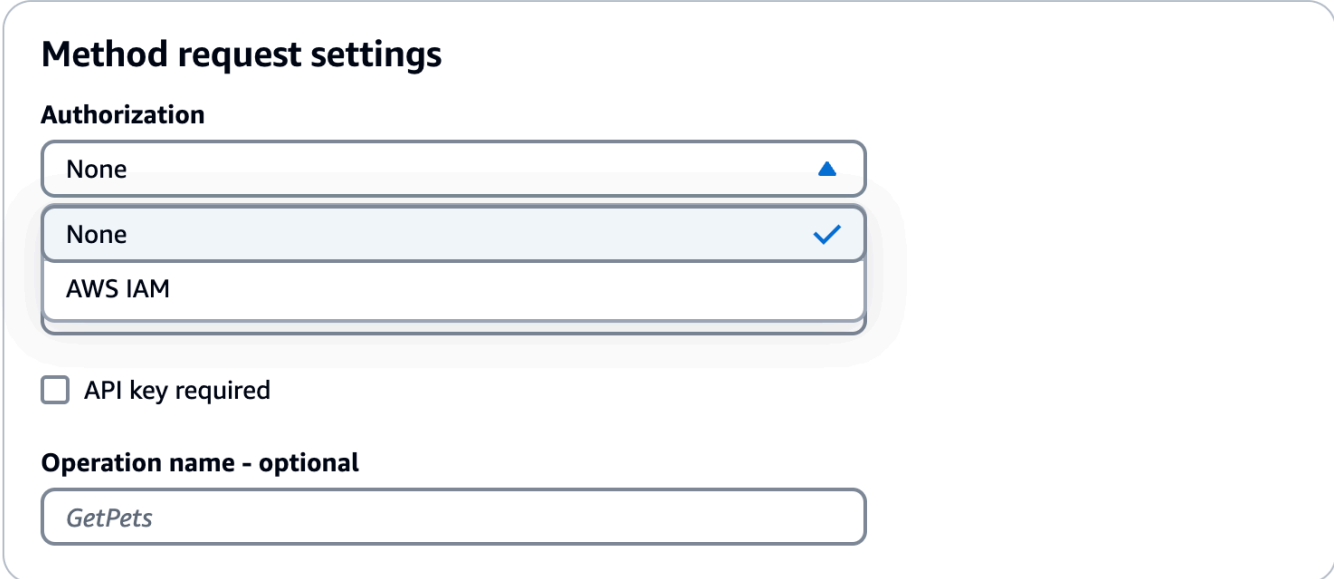
- Wählen Sie die `/`-Ressource aus und klicken Sie dann auf Methode erstellen.
- Wählen Sie als Methodentyp GET aus.
- Für Integrationstyp wählen Sie AWS-Service aus.
- Wählen Sie für den Ort aus AWS-Region, AWS-Region an dem Sie Ihren Amazon S3 S3-Bucket erstellt haben.
- Wählen Sie für AWS-Service Amazon Simple Storage Service aus.
- Lassen Sie die AWS -Subdomain leer.
- Für HTTP-Methode wählen Sie GET aus.
- Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus.

Wenn der Pfad überschrieben wird, leitet API Gateway die Client-Anfrage an Amazon S3 als die entsprechende [Amazon S3-REST-API-Path-Style-Anfrage](#) weiter, worin eine Amazon S3-Ressource durch den Ressourcen-Pfad des `s3-host-name/bucket/key`-Musters ausgedrückt wird. API Gateway legt den `s3-host-name` fest und gibt den Client, der in `bucket` und `key` spezifiziert ist, vom Client an Amazon S3 weiter.

9. Geben Sie für Pfadüberschreibung / ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **APIGatewayS3ProxyPolicy** ein.
11. Klicken Sie auf Einstellungen für Methodenanfragen.

Mit den Einstellungen für die Methodenanfragen steuern Sie, wer diese Methode Ihrer API aufrufen kann.

12. Wählen Sie im Dropdown-Menü `AWS_IAM` für Autorisierung aus.



Method request settings

Authorization

- None
- None ✓
- AWS IAM

☐ API key required

Operation name - optional

GetPets

13. Wählen Sie Methode erstellen aus.

Mit dieser Einrichtung wird die Frontend-Anforderung GET `https://your-api-host/stage/` in das GET `https://your-s3-host/` am Backend integriert.

Sie deklarieren die 200-, 400- und 500-Antworten unter Methodenantwort, damit Ihre API erfolgreiche Antworten und Ausnahmen ordnungsgemäß an den Aufrufer zurückgibt. Die Standard-Zuweisung für 200-Antworten wird verwendet, sodass hier nicht deklarierte Backend-Antworten des Statuscodes an den Aufrufer als 200-Antworten zurückgegeben werden.

Deklarieren der Antworttypen für die **GET** /-Methode

1. Klicken Sie auf der Registerkarte Methodenantwort unter Antwort 200 auf Bearbeiten.
2. Klicken Sie auf Header hinzufügen und gehen Sie wie folgt vor:
 - a. Für Header-Name geben Sie **Content-Type** ein.
 - b. Wählen Sie Add header.

Wiederholen Sie diese Schritte, um einen **Timestamp**-Header und einen **Content-Length**-Header zu erstellen.

3. Wählen Sie Speichern.
4. Klicken Sie auf der Registerkarte Methodenantwort unter Methodenantworten auf Antwort erstellen.
5. Geben Sie als HTTP-Statuscode 400 ein.

Für diese Antwort legen Sie keine Header fest.

6. Wählen Sie Speichern.
7. Wiederholen Sie die folgenden Schritte, um die Antwort 500 zu erstellen.

Für diese Antwort legen Sie keine Header fest.

Da die erfolgreiche Integrationsantwort von Amazon S3 die Bucket-Liste als XML-Nutzlast, und die standardmäßige Methodenantwort von API Gateway eine JSON-Nutzlast zurückgibt, muss der Parameterwert des Backend-Content-Type-Headers dem Frontend-Gegenstück zugewiesen werden. Andernfalls erhält der Client `application/json` als Content-Type, wenn der Antworttext eigentlich eine XML-Zeichenfolge ist. Das folgende Verfahren zeigt, wie diese Einrichtung erfolgt. Weiterhin sollten Sie dem Client noch andere Header-Parameter anzeigen, z. B. Datum und Länge des Inhalts.

So richten Sie Antwort-Header-Zuordnungen für die GET-Methode ein

1. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
2. Geben Sie für den Header Content-Length den Wert **integration.response.header.Content-Length** als Zuordnungswert ein.
3. Geben Sie für den Header Content-Type den Wert **integration.response.header.Content-Type** als Zuordnungswert ein.

- Geben Sie für den Header Timestamp den Wert **integration.response.header.Date** als Zuordnungswert ein.
- Wählen Sie Speichern. Das Ergebnis sollte in etwa wie folgt aussehen:

<
Method request
Integration request
Integration response
Met
>

Integration responses

Create response

Default - Response

EditDelete

HTTP status regex [Info](#)
-

Content handling [Learn more](#) [↗](#)
Passthrough

Method response status code
200

Default mapping
True

Header mappings (3)
< 1 >

Name ▲	Mapping value ▼
method.response.header.Content-Length	integration.response.header.Content-Length
method.response.header.Content-Type	integration.response.header.Content-Type
method.response.header.Timestamp	integration.response.header.Date

Mapping templates (0)

No templates

You don't have any mapping templates.

- Klicken Sie auf der Registerkarte Integrationsantwort unter Integrationsantworten auf Antwort erstellen.
- Machen Sie für HTTP status regex (HTTP-Status-RegEx) den Eintrag **4\d{2}**. Dadurch werden alle 4xx-HTTP-Antwortstatuscodes der Methodenantwort zugeordnet.

8. Wählen Sie für Statuscode der Methodenantwort **400** aus.
9. Wählen Sie Erstellen aus.
10. Wiederholen Sie die folgenden Schritte, um eine Integrationsantwort für die 500-Methodenantwort zu erstellen. Machen Sie für HTTP status regex (HTTP-Status-RegEx) den Eintrag **5\d{2}**.

Zur Übung sollten Sie nun die API testen, die Sie bisher konfiguriert haben.

So testen Sie die **GET** /-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie Test aus. Das Ergebnis sollte wie in der folgenden Abbildung aussehen:

Method request

Integration request

Integration response

Method response

Test

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Query strings

`param1=value1&param2=value2`

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

`header1:value1`
`header2:value2`

Client certificate

No client certificates have been generated.

Test

/ - GET method test results

Request

/

Latency ms

74

Status

200

Response body

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner>
<ID>abcd123456789abcd</ID><DisplayName>weizhang</DisplayName></Owner><Buckets><Bucket>
<Name>amzn-s3-demo-bucket</Name><CreationDate>2023-06-29T17:52:42.000</CreationDate>
</Bucket><Bucket><Name>amzn-s3-demo-bucket1</Name><CreationDate>2023-02-
```

Bereitstellen von API-Methoden, um auf einen Amazon S3-Bucket zuzugreifen

Wenn Sie mit einem Amazon-S3-Bucket arbeiten, legen Sie die GET-Methode in der `/`-Ressource zum Auflisten von Objekten in einem Bucket offen. Die Anweisungen sind ähnlich wie diejenigen, die unter beschrieben sind [Bereitstellen einer API-Methode zur Auflistung der Amazon S3-Buckets des Aufrufers](#). Sie können die Beispiel-API hier für weitere Methoden importieren, [OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy](#).

Exponieren der GET-Methoden in einer Ordnerressource

1. Wählen Sie die `/`-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp GET aus.

3. Für Integrationstyp wählen Sie AWS-Service aus.
4. Wählen Sie für den Ort aus AWS-Region, AWS-Region an dem Sie Ihren Amazon S3 S3-Bucket erstellt haben.
5. Wählen Sie für AWS-Service Amazon Simple Storage Service aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Für HTTP-Methode wählen Sie GET aus.
8. Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus.
9. Geben Sie für Pfadüberschreibung **{bucket}** ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **APIGatewayS3ProxyPolicy** ein.
11. Wählen Sie Methode erstellen aus.

Der `{folder}`-Pfadparameter wird in der Amazon-S3-Endpunkt-URL festgelegt. Der `{folder}`-Pfadparameter der Methodenanforderung muss dem `{bucket}`-Pfadparameter der Integrationsanforderung zugeordnet werden.

Zuordnen von **{folder}** zu **{bucket}**

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie URL-Pfadparameter aus und klicken Sie dann auf Pfadparameter hinzufügen.
3. Geben Sie unter Name **bucket** ein.
4. Geben Sie für Zugeordnet von **method.request.path.folder** ein.
5. Wählen Sie Speichern.

Als Nächstes testen Sie Ihre API.

Testen der **/ {folder} GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Geben Sie den Namen Ihres Buckets unter Pfad für Ordner ein.
3. Wählen Sie Test aus.

Das Testergebnis wird eine Liste der Objekte in Ihrem Bucket enthalten.

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Path

folder

Query strings

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

Client certificate

No client certificates have been generated.

Test

ⓘ /{folder} - GET method test results

Request

/amzn-s3-demo-bucket

Latency ms

59

Status

200

Response body

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Name>amzn-s3-demo-
bucket</Name><Prefix></Prefix><Marker></Marker><MaxKeys>1000</MaxKeys>
<IsTruncated>>false</IsTruncated><Contents><Key>test.txt</Key><LastModified>2023-10-
```

Bereitstellen von API-Methoden für den Zugriff auf ein Amazon S3-Objekt in einem Bucket

Amazon S3 unterstützt GET-, DELETE-, HEAD-, OPTIONS-, POST- und PUT-Aktionen für den Zugriff auf und die Verwaltung von Objekte(n) in einem bestimmten Bucket. In diesem Tutorial exponieren Sie eine GET-Methode für die `{folder}/{item}`-Ressource, um eine Abbildung aus einem Bucket abzurufen. Weitere Nutzungsbeispiele der `{folder}/{item}` Ressource finden Sie in der Beispiel-API, [OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy](#).

Exponieren der GET-Methode in einer item-Ressource

1. Wählen Sie die `/item`-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp GET aus.
3. Für Integrationstyp wählen Sie AWS-Service aus.

4. Wählen Sie für den Ort aus AWS-Region, AWS-Region an dem Sie Ihren Amazon S3 S3-Bucket erstellt haben.
5. Wählen Sie für AWS-Service Amazon Simple Storage Service aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Für HTTP-Methode wählen Sie GET aus.
8. Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus.
9. Geben Sie für Pfadüberschreibung `{bucket}/{object}` ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **APIGatewayS3ProxyPolicy** ein.
11. Wählen Sie Methode erstellen aus.

Die Pfadparameter `{folder}` und `{item}` werden in der Amazon-S3-Endpunkt-URL festgelegt. Der -Pfadparameter der Methodenanforderung muss dem -Pfadparameter der Integrationsanforderung zugeordnet werden.

In diesem Schritt führen Sie folgende Aufgaben aus:

- Ordnen Sie den Pfadparameter `{folder}` der Methodenanforderung dem Pfadparameter `{bucket}` der Integrationsanforderung zu.
- Ordnen Sie den Pfadparameter `{item}` der Methodenanforderung dem Pfadparameter `{object}` der Integrationsanforderung zu.

Zuordnen von **{folder}** zu **{bucket}** und **{item}** zu **{object}**

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Pfadparameter.
3. Klicken Sie auf Pfadparameter hinzufügen.
4. Geben Sie unter Name **bucket** ein.
5. Geben Sie für Zugeordnet von **method.request.path.folder** ein.
6. Klicken Sie auf Pfadparameter hinzufügen.
7. Geben Sie unter Name **object** ein.
8. Geben Sie für Zugeordnet von **method.request.path.item** ein.
9. Wählen Sie Speichern.

Testen der `/{{folder}}/{{object}}` GET-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Geben Sie den Namen Ihres Buckets unter Pfad für Ordner ein.
3. Geben Sie unter Pfad, Element den Namen eines Elements ein.
4. Wählen Sie Test aus.

Der Antworttext enthält den Inhalt des Elements.

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Path

folder

item

Query strings

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

Client certificate

No client certificates have been generated. ▼

Test

 **/{folder}}/{item}} - GET method test results**

Request	Latency ms	Status
/amzn-s3-demo-bucket/test.txt	61	200
Response body		
Hello World		

Die Anforderung gibt den Klartext („Hello world“) als Inhalt der angegebenen Datei (test.txt) in dem entsprechenden Amazon-S3-Bucket (amzn-s3-demo-bucket) zurück.

Um Binärdateien, die in API Gateway als irgendetwas anderes als utf-8-codierter JSON-Inhalt betrachtet werden, herunter- oder hochzuladen, sind zusätzliche API-Einstellungen erforderlich. Dies kann wie folgt beschrieben werden:

Herunterladen oder Hochladen von Binärdateien von S3

1. Registrieren Sie die Medientypen der betroffenen Datei bei den APIs `binaryMediaTypes`. Sie können Sie von der Konsole aus erledigen:
 - a. Wählen Sie API-Einstellungen für die API aus.
 - b. Klicken Sie unter Binäre Medientypen auf Medientypen verwalten.
 - c. Klicken Sie auf Binären Medientyp hinzufügen und geben Sie dann den erforderlichen Medientyp ein, z. B. `image/png`.
 - d. Wählen Sie zum Speichern der Einstellung `Save Changes` (Änderungen speichern) aus.
2. Fügen Sie der Methodenanforderung den Header `Content-Type` and/or `Accept` (zum Hochladen) (zum Herunterladen) hinzu, damit der Client den erforderlichen binären Medientyp angeben und ihn der Integrationsanforderung zuordnen muss.
3. Setzen Sie `Content Handling` in der Integrationsanfrage auf `Passthrough` (für das Hochladen), und in einer Integrationsantwort (für das Herunterladen). Stellen Sie sicher, dass keine Mapping-Vorlage für den betroffenen Inhaltstyp definiert ist. Weitere Informationen finden Sie unter [the section called “Datentransformationen”](#).

Die maximale Größe der Nutzlast ist 10 MB. Siehe [Kontingente für die Konfiguration und Ausführung einer REST-API in API Gateway](#).

Stellen Sie sicher, dass für Dateien in Amazon S3 die richtigen Inhaltstypen als Metadaten der Dateien hinzugefügt wurden. Bei streamfähigen Medieninhalten muss den Metadaten möglicherweise auch `Content-Disposition:inline` hinzugefügt werden.

Weitere Informationen zur Unterstützung von Binärdateien in API Gateway finden Sie in [Inhaltstypkonvertierungen in API Gateway](#).

OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy

Die folgenden OpenAPI-Definitionen beschreiben eine API, die als Amazon-S3-Proxy operiert. Diese API enthält mehr Amazon-S3-Operationen als die API, die Sie im Tutorial erstellt haben. Die folgenden Methoden sind in den OpenAPI-Definitionen exponiert:

- Bereitstellung von GET bei der Stammressource der API, um [alle Amazon S3-Buckets eines Aufrufers aufzulisten](#).
- Bereitstellung von GET bei einer Folder-Ressource, um [eine Liste aller Objekte in einem Amazon S3-Bucket anzuzeigen](#).
- Bereitstellung von PUT bei einer Folder-Ressource, um [einen Bucket zu Amazon S3 hinzuzufügen](#).
- Bereitstellung von DELETE bei einer Folder-Ressource, um [einen Bucket aus Amazon S3 zu entfernen](#).
- Machen Sie GET auf einer Folder/Item Ressource verfügbar, um [ein Objekt aus einem Amazon S3 S3-Bucket anzuzeigen oder herunterzuladen](#).
- Machen Sie PUT auf einer Folder/Item Ressource verfügbar, um [ein Objekt in einen Amazon S3 S3-Bucket hochzuladen](#).
- Machen Sie HEAD auf einer Folder/Item Ressource verfügbar, [um Objektmetadaten in einem Amazon S3 S3-Bucket abzurufen](#).
- Machen Sie DELETE für eine Folder/Item Ressource verfügbar, um [ein Objekt aus einem Amazon S3 S3-Bucket zu entfernen](#).

Anweisung zum Importieren einer API mithilfe der OpenAPI-Definition finden Sie unter [REST-APIs mit OpenAPI in API Gateway erstellen](#).

Eine Anleitung zur Erstellung einer ähnlichen API finden Sie unter [Tutorial: REST-API als AmazonS3-Proxy erstellen](#).

Informationen zum Aufrufen dieser API mit [Postman](#), das die AWS IAM-Autorisierung unterstützt, finden Sie unter. [Aufrufen der API mit einem REST-API-Client](#)

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-13T23:04:43Z",
    "title": "MyS3"
  },
  "host": "9gn28ca086.execute-api.{region}.amazonaws.com",
  "basePath": "/S3",
  "schemes": [
    "https"
  ],
```

```
"paths": {
  "/": {
    "get": {
      "produces": [
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          },
          "headers": {
            "Content-Length": {
              "type": "string"
            },
            "Timestamp": {
              "type": "string"
            },
            "Content-Type": {
              "type": "string"
            }
          }
        },
        "400": {
          "description": "400 response"
        },
        "500": {
          "description": "500 response"
        }
      },
      "security": [
        {
          "sigv4": []
        }
      ],
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
          "4\\d{2}": {
            "statusCode": "400"
          },
          "default": {
            "statusCode": "200",
```

```

        "responseParameters": {
            "method.response.header.Content-Type":
"integration.response.header.Content-Type",
            "method.response.header.Content-Length":
"integration.response.header.Content-Length",
            "method.response.header.Timestamp":
"integration.response.header.Date"
        }
    },
    "5\\d{2}": {
        "statusCode": "500"
    }
},
"uri": "arn:aws:apigateway:us-west-2:s3:path//",
"passthroughBehavior": "when_no_match",
"httpMethod": "GET",
"type": "aws"
}
},
"/{folder}": {
    "get": {
        "produces": [
            "application/json"
        ],
        "parameters": [
            {
                "name": "folder",
                "in": "path",
                "required": true,
                "type": "string"
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "schema": {
                    "$ref": "#/definitions/Empty"
                },
                "headers": {
                    "Content-Length": {
                        "type": "string"
                    },
                    "Date": {

```

```

        "type": "string"
      },
      "Content-Type": {
        "type": "string"
      }
    }
  },
  "400": {
    "description": "400 response"
  },
  "500": {
    "description": "500 response"
  }
},
"security": [
  {
    "sigv4": []
  }
],
"x-amazon-apigateway-integration": {
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "responses": {
    "4\\d{2}": {
      "statusCode": "400"
    },
    "default": {
      "statusCode": "200",
      "responseParameters": {
        "method.response.header.Content-Type":
"integration.response.header.Content-Type",
        "method.response.header.Date": "integration.response.header.Date",
        "method.response.header.Content-Length":
"integration.response.header.content-length"
      }
    },
    "5\\d{2}": {
      "statusCode": "500"
    }
  },
  "requestParameters": {
    "integration.request.path.bucket": "method.request.path.folder"
  },
  "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
  "passthroughBehavior": "when_no_match",

```

```
    "httpMethod": "GET",
    "type": "aws"
  },
  "put": {
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "Content-Type",
        "in": "header",
        "required": false,
        "type": "string"
      },
      {
        "name": "folder",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "200 response",
        "schema": {
          "$ref": "#/definitions/Empty"
        },
        "headers": {
          "Content-Length": {
            "type": "string"
          },
          "Content-Type": {
            "type": "string"
          }
        }
      },
      "400": {
        "description": "400 response"
      },
      "500": {
        "description": "500 response"
      }
    }
  },
}
```

```

    "security": [
      {
        "sigv4": []
      }
    ],
    "x-amazon-apigateway-integration": {
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "responses": {
        "4\\d{2}": {
          "statusCode": "400"
        },
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.Content-Type":
"integration.response.header.Content-Type",
            "method.response.header.Content-Length":
"integration.response.header.Content-Length"
          }
        },
        "5\\d{2}": {
          "statusCode": "500"
        }
      },
      "requestParameters": {
        "integration.request.path.bucket": "method.request.path.folder",
        "integration.request.header.Content-Type":
"method.request.header.Content-Type"
      },
      "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
      "passthroughBehavior": "when_no_match",
      "httpMethod": "PUT",
      "type": "aws"
    }
  },
  "delete": {
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "folder",
        "in": "path",
        "required": true,

```

```

        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "200 response",
        "schema": {
          "$ref": "#/definitions/Empty"
        },
        "headers": {
          "Date": {
            "type": "string"
          },
          "Content-Type": {
            "type": "string"
          }
        }
      },
      "400": {
        "description": "400 response"
      },
      "500": {
        "description": "500 response"
      }
    },
    "security": [
      {
        "sigv4": []
      }
    ],
    "x-amazon-apigateway-integration": {
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "responses": {
        "4\\d{2}": {
          "statusCode": "400"
        },
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.Content-Type":
"integration.response.header.Content-Type",
            "method.response.header.Date": "integration.response.header.Date"
          }
        }
      }
    },

```

```

        "5\\d{2}": {
            "statusCode": "500"
        }
    },
    "requestParameters": {
        "integration.request.path.bucket": "method.request.path.folder"
    },
    "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "DELETE",
    "type": "aws"
}
}
},
"/{folder}/{item}": {
    "get": {
        "produces": [
            "application/json"
        ],
        "parameters": [
            {
                "name": "item",
                "in": "path",
                "required": true,
                "type": "string"
            },
            {
                "name": "folder",
                "in": "path",
                "required": true,
                "type": "string"
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "schema": {
                    "$ref": "#/definitions/Empty"
                },
                "headers": {
                    "content-type": {
                        "type": "string"
                    },
                    "Content-Type": {

```



```

        "type": "string"
      }
    }
  },
  "400": {
    "description": "400 response"
  },
  "500": {
    "description": "500 response"
  }
},
"security": [
  {
    "sigv4": []
  }
],
"x-amazon-apigateway-integration": {
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "responses": {
    "4\\d{2}": {
      "statusCode": "400"
    },
    "default": {
      "statusCode": "200",
      "responseParameters": {
        "method.response.header.content-type":
"integration.response.header.content-type",
        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
      }
    },
    "5\\d{2}": {
      "statusCode": "500"
    }
  },
  "requestParameters": {
    "integration.request.path.object": "method.request.path.item",
    "integration.request.path.bucket": "method.request.path.folder"
  },
  "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
  "passthroughBehavior": "when_no_match",
  "httpMethod": "GET",
  "type": "aws"
}

```

```
},
"head": {
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "name": "item",
      "in": "path",
      "required": true,
      "type": "string"
    },
    {
      "name": "folder",
      "in": "path",
      "required": true,
      "type": "string"
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Empty"
      },
      "headers": {
        "Content-Length": {
          "type": "string"
        },
        "Content-Type": {
          "type": "string"
        }
      }
    },
    "400": {
      "description": "400 response"
    },
    "500": {
      "description": "500 response"
    }
  },
  "security": [
    {
      "sigv4": []
    }
  ]
}
```

```

    }
  ],
  "x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
      "4\\d{2}": {
        "statusCode": "400"
      },
      "default": {
        "statusCode": "200",
        "responseParameters": {
          "method.response.header.Content-Type":
"integration.response.header.Content-Type",
          "method.response.header.Content-Length":
"integration.response.header.Content-Length"
        }
      },
      "5\\d{2}": {
        "statusCode": "500"
      }
    },
    "requestParameters": {
      "integration.request.path.object": "method.request.path.item",
      "integration.request.path.bucket": "method.request.path.folder"
    },
    "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "HEAD",
    "type": "aws"
  }
},
"put": {
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "name": "Content-Type",
      "in": "header",
      "required": false,
      "type": "string"
    },
    {
      "name": "item",

```

```
        "in": "path",
        "required": true,
        "type": "string"
    },
    {
        "name": "folder",
        "in": "path",
        "required": true,
        "type": "string"
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "schema": {
            "$ref": "#/definitions/Empty"
        },
        "headers": {
            "Content-Length": {
                "type": "string"
            },
            "Content-Type": {
                "type": "string"
            }
        }
    },
    "400": {
        "description": "400 response"
    },
    "500": {
        "description": "500 response"
    }
},
"security": [
    {
        "sigv4": []
    }
],
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "4\\d{2}": {
            "statusCode": "400"
        }
    },

```

```

        "default": {
            "statusCode": "200",
            "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type",
                "method.response.header.Content-Length":
"integration.response.header.Content-Length"
            }
        },
        "5\\d{2}": {
            "statusCode": "500"
        }
    },
    "requestParameters": {
        "integration.request.path.object": "method.request.path.item",
        "integration.request.path.bucket": "method.request.path.folder",
        "integration.request.header.Content-Type":
"method.request.header.Content-Type"
    },
    "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "PUT",
    "type": "aws"
}
},
"delete": {
    "produces": [
        "application/json"
    ],
    "parameters": [
        {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
        },
        {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
        }
    ],
    "responses": {

```

```
"200": {
  "description": "200 response",
  "schema": {
    "$ref": "#/definitions/Empty"
  },
  "headers": {
    "Content-Length": {
      "type": "string"
    },
    "Content-Type": {
      "type": "string"
    }
  }
},
"400": {
  "description": "400 response"
},
"500": {
  "description": "500 response"
}
},
"security": [
  {
    "sigv4": []
  }
],
"x-amazon-apigateway-integration": {
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "responses": {
    "4\\d{2}": {
      "statusCode": "400"
    },
    "default": {
      "statusCode": "200"
    },
    "5\\d{2}": {
      "statusCode": "500"
    }
  },
  "requestParameters": {
    "integration.request.path.object": "method.request.path.item",
    "integration.request.path.bucket": "method.request.path.folder"
  },
  "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
```

```

        "passthroughBehavior": "when_no_match",
        "httpMethod": "DELETE",
        "type": "aws"
    }
}
},
"securityDefinitions": {
    "sigv4": {
        "type": "apiKey",
        "name": "Authorization",
        "in": "header",
        "x-amazon-apigateway-authtype": "awsSigv4"
    }
},
"definitions": {
    "Empty": {
        "type": "object",
        "title": "Empty Schema"
    }
}
}

```

OpenAPI 3.0

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "MyS3",
    "version" : "2016-10-13T23:04:43Z"
  },
  "servers" : [ {
    "url" : "https://9gn28ca086.execute-api.{region}.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "S3"
      }
    }
  } ],
  "paths" : {
    "/{folder}" : {
      "get" : {
        "parameters" : [ {

```

```
    "name" : "folder",
    "in" : "path",
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  } ],
  "responses" : {
    "400" : {
      "description" : "400 response",
      "content" : { }
    },
    "500" : {
      "description" : "500 response",
      "content" : { }
    },
    "200" : {
      "description" : "200 response",
      "headers" : {
        "Content-Length" : {
          "schema" : {
            "type" : "string"
          }
        },
        "Date" : {
          "schema" : {
            "type" : "string"
          }
        },
        "Content-Type" : {
          "schema" : {
            "type" : "string"
          }
        }
      },
      "content" : {
        "application/json" : {
          "schema" : {
            "$ref" : "#/components/schemas/Empty"
          }
        }
      }
    }
  }
},
```



```

"x-amazon-apigateway-integration" : {
  "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "httpMethod" : "GET",
  "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
  "responses" : {
    "4\\d{2}" : {
      "statusCode" : "400"
    },
    "default" : {
      "statusCode" : "200",
      "responseParameters" : {
        "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
        "method.response.header.Date" : "integration.response.header.Date",
        "method.response.header.Content-Length" :
"integration.response.header.content-length"
      }
    },
    "5\\d{2}" : {
      "statusCode" : "500"
    }
  },
  "requestParameters" : {
    "integration.request.path.bucket" : "method.request.path.folder"
  },
  "passthroughBehavior" : "when_no_match",
  "type" : "aws"
},
"put" : {
  "parameters" : [ {
    "name" : "Content-Type",
    "in" : "header",
    "schema" : {
      "type" : "string"
    }
  }, {
    "name" : "folder",
    "in" : "path",
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  }
]
},

```

```

"responses" : {
  "400" : {
    "description" : "400 response",
    "content" : { }
  },
  "500" : {
    "description" : "500 response",
    "content" : { }
  },
  "200" : {
    "description" : "200 response",
    "headers" : {
      "Content-Length" : {
        "schema" : {
          "type" : "string"
        }
      },
      "Content-Type" : {
        "schema" : {
          "type" : "string"
        }
      }
    },
    "content" : {
      "application/json" : {
        "schema" : {
          "$ref" : "#/components/schemas/Empty"
        }
      }
    }
  }
},
"x-amazon-apigateway-integration" : {
  "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "httpMethod" : "PUT",
  "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
  "responses" : {
    "4\\d{2}" : {
      "statusCode" : "400"
    },
    "default" : {
      "statusCode" : "200",
      "responseParameters" : {

```

```

        "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
        "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
    }
},
    "5\\d{2}" : {
        "statusCode" : "500"
    }
},
    "requestParameters" : {
        "integration.request.path.bucket" : "method.request.path.folder",
        "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "aws"
}
},
"delete" : {
    "parameters" : [ {
        "name" : "folder",
        "in" : "path",
        "required" : true,
        "schema" : {
            "type" : "string"
        }
    } ],
    "responses" : {
        "400" : {
            "description" : "400 response",
            "content" : { }
        },
        "500" : {
            "description" : "500 response",
            "content" : { }
        },
        "200" : {
            "description" : "200 response",
            "headers" : {
                "Date" : {
                    "schema" : {
                        "type" : "string"
                    }
                }
            }
        }
    }
}

```

```

        },
        "Content-Type" : {
            "schema" : {
                "type" : "string"
            }
        }
    },
    "content" : {
        "application/json" : {
            "schema" : {
                "$ref" : "#/components/schemas/Empty"
            }
        }
    }
}
},
"x-amazon-apigateway-integration" : {
    "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "httpMethod" : "DELETE",
    "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
    "responses" : {
        "4\\d{2}" : {
            "statusCode" : "400"
        },
        "default" : {
            "statusCode" : "200",
            "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Date" : "integration.response.header.Date"
            }
        },
        "5\\d{2}" : {
            "statusCode" : "500"
        }
    },
    "requestParameters" : {
        "integration.request.path.bucket" : "method.request.path.folder"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "aws"
}
},
},
},

```

```
"/{folder}/{item}" : {
  "get" : {
    "parameters" : [ {
      "name" : "item",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    }, {
      "name" : "folder",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    } ],
    "responses" : {
      "400" : {
        "description" : "400 response",
        "content" : { }
      },
      "500" : {
        "description" : "500 response",
        "content" : { }
      },
      "200" : {
        "description" : "200 response",
        "headers" : {
          "content-type" : {
            "schema" : {
              "type" : "string"
            }
          },
          "Content-Type" : {
            "schema" : {
              "type" : "string"
            }
          }
        },
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/Empty"
            }
          }
        }
      }
    }
  }
}
```

```

        }
      }
    }
  },
  "x-amazon-apigateway-integration" : {
    "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "httpMethod" : "GET",
    "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
    "responses" : {
      "4\\d{2}" : {
        "statusCode" : "400"
      },
      "default" : {
        "statusCode" : "200",
        "responseParameters" : {
          "method.response.header.content-type" :
"integration.response.header.content-type",
          "method.response.header.Content-Type" :
"integration.response.header.Content-Type"
        }
      },
      "5\\d{2}" : {
        "statusCode" : "500"
      }
    },
    "requestParameters" : {
      "integration.request.path.object" : "method.request.path.item",
      "integration.request.path.bucket" : "method.request.path.folder"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "aws"
  }
},
"put" : {
  "parameters" : [ {
    "name" : "Content-Type",
    "in" : "header",
    "schema" : {
      "type" : "string"
    }
  }
], {
  "name" : "item",
  "in" : "path",

```

```
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  }, {
    "name" : "folder",
    "in" : "path",
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  } ],
  "responses" : {
    "400" : {
      "description" : "400 response",
      "content" : { }
    },
    "500" : {
      "description" : "500 response",
      "content" : { }
    },
    "200" : {
      "description" : "200 response",
      "headers" : {
        "Content-Length" : {
          "schema" : {
            "type" : "string"
          }
        },
        "Content-Type" : {
          "schema" : {
            "type" : "string"
          }
        }
      },
      "content" : {
        "application/json" : {
          "schema" : {
            "$ref" : "#/components/schemas/Empty"
          }
        }
      }
    }
  }
},
```

```

"x-amazon-apigateway-integration" : {
  "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "httpMethod" : "PUT",
  "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
  "responses" : {
    "4\\d{2}" : {
      "statusCode" : "400"
    },
    "default" : {
      "statusCode" : "200",
      "responseParameters" : {
        "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
        "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
      }
    },
    "5\\d{2}" : {
      "statusCode" : "500"
    }
  },
  "requestParameters" : {
    "integration.request.path.object" : "method.request.path.item",
    "integration.request.path.bucket" : "method.request.path.folder",
    "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
  },
  "passthroughBehavior" : "when_no_match",
  "type" : "aws"
}
}, {
  "delete" : {
    "parameters" : [ {
      "name" : "item",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    }
  ], {
    "name" : "folder",
    "in" : "path",
    "required" : true,
    "schema" : {

```



```

        "type" : "string"
      }
    } ],
    "responses" : {
      "400" : {
        "description" : "400 response",
        "content" : { }
      },
      "500" : {
        "description" : "500 response",
        "content" : { }
      },
      "200" : {
        "description" : "200 response",
        "headers" : {
          "Content-Length" : {
            "schema" : {
              "type" : "string"
            }
          },
          "Content-Type" : {
            "schema" : {
              "type" : "string"
            }
          }
        },
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/Empty"
            }
          }
        }
      }
    },
    "x-amazon-apigateway-integration" : {
      "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "httpMethod" : "DELETE",
      "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
      "responses" : {
        "4\\d{2}" : {
          "statusCode" : "400"
        },
        "default" : {

```

```

        "statusCode" : "200"
      },
      "5\\d{2}" : {
        "statusCode" : "500"
      }
    },
    "requestParameters" : {
      "integration.request.path.object" : "method.request.path.item",
      "integration.request.path.bucket" : "method.request.path.folder"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "aws"
  }
},
"head" : {
  "parameters" : [ {
    "name" : "item",
    "in" : "path",
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  }, {
    "name" : "folder",
    "in" : "path",
    "required" : true,
    "schema" : {
      "type" : "string"
    }
  } ],
  "responses" : {
    "400" : {
      "description" : "400 response",
      "content" : { }
    },
    "500" : {
      "description" : "500 response",
      "content" : { }
    },
    "200" : {
      "description" : "200 response",
      "headers" : {
        "Content-Length" : {
          "schema" : {

```

```

        "type" : "string"
      }
    },
    "Content-Type" : {
      "schema" : {
        "type" : "string"
      }
    }
  },
  "content" : {
    "application/json" : {
      "schema" : {
        "$ref" : "#/components/schemas/Empty"
      }
    }
  }
},
"x-amazon-apigateway-integration" : {
  "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "httpMethod" : "HEAD",
  "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
  "responses" : {
    "4\\d{2}" : {
      "statusCode" : "400"
    },
    "default" : {
      "statusCode" : "200",
      "responseParameters" : {
        "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
        "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
      }
    },
    "5\\d{2}" : {
      "statusCode" : "500"
    }
  },
  "requestParameters" : {
    "integration.request.path.object" : "method.request.path.item",
    "integration.request.path.bucket" : "method.request.path.folder"
  },
  "passthroughBehavior" : "when_no_match",

```

```
        "type" : "aws"
      }
    },
    "/": {
      "get": {
        "responses": {
          "400": {
            "description": "400 response",
            "content": { }
          },
          "500": {
            "description": "500 response",
            "content": { }
          },
          "200": {
            "description": "200 response",
            "headers": {
              "Content-Length": {
                "schema": {
                  "type": "string"
                }
              },
              "Timestamp": {
                "schema": {
                  "type": "string"
                }
              },
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration" : {
```

```

    "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "httpMethod" : "GET",
    "uri" : "arn:aws:apigateway:us-west-2:s3:path//",
    "responses" : {
      "4\\d{2}" : {
        "statusCode" : "400"
      },
      "default" : {
        "statusCode" : "200",
        "responseParameters" : {
          "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
          "method.response.header.Content-Length" :
"integration.response.header.Content-Length",
          "method.response.header.Timestamp" :
"integration.response.header.Date"
        }
      },
      "5\\d{2}" : {
        "statusCode" : "500"
      }
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "aws"
  }
}
},
"components" : {
  "schemas" : {
    "Empty" : {
      "title" : "Empty Schema",
      "type" : "object"
    }
  }
}
}
}

```

Aufrufen der API mit einem REST-API-Client

Als end-to-end Tutorial zeigen wir nun, wie die API mithilfe von [Postman](#) aufgerufen wird, das die IAM-Autorisierung unterstützt. AWS

So rufen Sie die Amazon S3-Proxy-API mit Postman auf

1. Stellen Sie die API bereit (oder erneut bereit). Notieren Sie die Basis-URL der API, die neben Invoke URL oben im Stage Editor angezeigt wird.
2. Starten Sie Postman.
3. Wählen Sie Autorisierung und anschließend AWS Signature. Geben Sie die Access Key ID und den Secret Access Key Ihres IAM-Benutzers in die jeweiligen AccessKeySecretKeyEingabefelder ein. Geben Sie im AWS-Region Textfeld Region die AWS Region ein, für die Ihre API bereitgestellt wird. Geben Sie `execute-api` im Eingabefeld Servicename ein.

Sie können ein Schlüsselpaar über die Registerkarte Security Credentials (Sicherheitsanmeldeinformationen) von Ihrem IAM-Benutzerkonto in der IAM Management Console erstellen.

4. So fügen Sie einen Bucket mit dem Namen `amzn-s3-demo-bucket` zu Ihrem Amazon S3-Konto in der Region `{region}` hinzu:
 - a. Wählen Sie in der Dropdown-Methodenliste PUT aus und geben Sie die Methoden-URL ein (`https://{api-id}.execute-api.{aws-region}.amazonaws.com/{stage}/{folder-name}`)
 - b. Geben Sie bei Content-Type den Header-Wert `application/xml` ein. Sie müssen möglicherweise vorhandene Header löschen, bevor Sie den Content-Type angeben können.
 - c. Wählen Sie das Menüelement Body aus, und geben Sie das folgende XML-Fragment als Anforderungstext ein:

```
<CreateBucketConfiguration>
  <LocationConstraint>{region}</LocationConstraint>
</CreateBucketConfiguration>
```

- d. Wählen Sie Senden aus, um die Anforderung zu senden. Bei Erfolg erhalten Sie die Antwort `200 OK` mit einer leeren Nutzlast.
5. Um zu einem Bucket eine Textdatei hinzuzufügen, befolgen Sie die obigen Anweisungen. Wenn Sie den Bucket-Namen **amzn-s3-demo-bucket** für `{folder}` und den Dateinamen für **Readme.txt** für `{item}` in der URL angeben und die Textzeichenfolge **Hello, World!** als Dateiinhalt (und damit als Anforderungsnutzlast) angeben, wird die Anforderung

```
PUT /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
```

```
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T062647Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-api/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
  Signature=ccadb877bdb0d395ca38cc47e18a0d76bb5eaf17007d11e40bf6fb63d28c705b
Cache-Control: no-cache
Postman-Token: 6135d315-9cc4-8af8-1757-90871d00847e

Hello, World!
```

Wenn alles einwandfrei verläuft, erhalten Sie die Antwort 200 OK mit einer leeren Nutzlast.

- Um den Inhalt der Datei `Readme.txt` abzurufen, die wir soeben zum `amzn-s3-demo-bucket`-Bucket hinzugefügt haben, erstellen wir eine GET-Anforderung, die wie folgt aussieht:

```
GET /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T063759Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
  Signature=ba09b72b585acf0e578e6ad02555c00e24b420b59025bc7bb8d3f7aed1471339
Cache-Control: no-cache
Postman-Token: d60fcb59-d335-52f7-0025-5bd96928098a
```

Bei Erfolg erhalten Sie die Antwort 200 OK mit der Textzeichenfolge `Hello, World!` als Nutzlast.

- Um Elemente im `amzn-s3-demo-bucket`-Bucket aufzulisten, senden Sie die folgende Anforderung:

```
GET /S3/amzn-s3-demo-bucket HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T064324Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
  Signature=4ac9bd4574a14e01568134fd16814534d9951649d3a22b3b0db9f1f5cd4dd0ac
Cache-Control: no-cache
Postman-Token: 9c43020a-966f-61e1-81af-4c49ad8d1392
```

Bei Erfolg erhalten Sie die Antwort 200 OK mit einer XML-Nutzlast mit einem einzelnen Element im angegebenen Bucket, es sei denn, Sie haben noch mehr Dateien zu dem Bucket hinzugefügt, bevor Sie diese Anforderung gesendet haben.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>apig-demo-5</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>Readme.txt</Key>
    <LastModified>2016-10-15T06:26:48.000Z</LastModified>
    <ETag>"65a8e27d8879283831b664bd8b7f0ad4"</ETag>
    <Size>13</Size>
    <Owner>
      <ID>06e4b09e9d...603addd12ee</ID>
      <DisplayName>user-name</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

Note

Um ein Bild hoch- oder herunterzuladen, müssen Sie für die Inhaltsbehandlung die Option CONVERT_TO_BINARY auswählen.

Tutorial: REST-API als Amazon Kinesis-Proxy erstellen

Auf dieser Seite wird beschrieben, wie eine REST-API durch Integration des AWS-Typs für den Zugriff auf Kinesis erstellt und konfiguriert wird.

 Note

Um Ihre API Gateway-API in Kinesis zu integrieren, müssen Sie eine Region wählen, in der sowohl das API Gateway als auch die Kinesis-Services verfügbar sind. Informationen zur Verfügbarkeit der Regionen finden Sie unter [Service-Endpunkte und Kontingente](#).

Zur Veranschaulichung erstellen wir eine Beispiel-API, so dass ein Client die folgenden Aufgaben ausführen kann:

1. Verfügbare Streams des Benutzers in Kinesis auflisten
2. Erstellen, Beschreiben oder Löschen eines bestimmten Streams
3. Lesen von Datensätzen aus einem oder Schreiben von Datensätzen in einen bestimmten Stream

Für die Ausführung der obigen Aufgaben stellt die API Methoden für verschiedene Ressourcen bereit, um Folgendes aufzurufen:

1. Die `ListStreams`-Aktion in Kinesis
2. Die Aktion `CreateStream`, `DescribeStream` oder `DeleteStream`
3. Die `GetRecords`- oder `PutRecords`-Aktion (einschließlich `PutRecord`) in Kinesis

Genauer gesagt wird die API wie folgt erstellt:

- Machen Sie eine HTTP-GET-Methode für die `/streams` API-Ressource verfügbar und integrieren Sie die Methode in die [ListStreams](#)Aktion in Kinesis, um die Streams im Konto des Aufrufers aufzulisten.
- Machen Sie eine HTTP-POST-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die [CreateStream](#)Aktion in Kinesis, um einen benannten Stream im Konto des Aufrufers zu erstellen.
- Machen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die [DescribeStream](#)Aktion in Kinesis, um einen benannten Stream im Konto des Aufrufers zu beschreiben.
- Machen Sie eine HTTP-DELETE-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die [DeleteStream](#)Aktion in Kinesis, um einen Stream im Konto des Aufrufers zu löschen.

- Machen Sie eine HTTP-PUT-Methode für die `/streams/{stream-name}/record` API-Ressource verfügbar und integrieren Sie die Methode in die [PutRecord](#)Aktion in Kinesis. Dadurch kann der Client einen einzelnen Datensatz zum benannten Stream hinzufügen.
- Machen Sie eine HTTP-PUT-Methode für die `/streams/{stream-name}/records` API-Ressource verfügbar und integrieren Sie die Methode in die [PutRecords](#)Aktion in Kinesis. Dadurch kann der Client eine Liste mit Datensätzen zum benannten Stream hinzufügen.
- Stellen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}/records` API-Ressource bereit und integrieren Sie die Methode in die [GetRecords](#)Aktion in Kinesis. Auf diese Weise kann der Client Datensätze im benannten Stream mit einem angegebenen gemeinsamen Iterator auflisten. Ein gemeinsamer Iterator gibt die gemeinsame Position an, ab der Datensätze sequenziell ausgelesen werden sollen.
- Stellen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}/sharditerator` API-Ressource bereit und integrieren Sie die Methode in die [GetShardIterator](#)Aktion in Kinesis. Diese Hilfsmethode muss der `ListStreams`-Aktion in Kinesis zur Verfügung gestellt werden.

Sie können die hier vorgestellten Anweisungen auf andere Kinesis-Aktionen anwenden. Die vollständige Liste der Kinesis-Aktionen finden Sie unter [Amazon Kinesis API-Referenz](#).

Anstatt die API Gateway-Konsole zur Erstellung der Beispiel-API zu verwenden, können Sie die Beispiel-API mit der API Gateway [Import API](#) in API Gateway importieren. Informationen zur Verwendung der Import-API finden Sie unter [REST-APIs mit OpenAPI in API Gateway erstellen](#).

Erstellen Sie eine IAM-Rolle und Richtlinien für den Zugriff der API auf Kinesis

Einer IAM-Rolle müssen entsprechende IAM-Richtlinien zugeordnet sein, damit die API Kinesis-Aktionen aufrufen kann. In dieser Übung erstellen Sie eine neue IAM-Rolle.

So erstellen Sie die Ausführungsrolle AWS für den Service-Proxy

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie Roles.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS Dienst aus, wählen Sie dann API Gateway aus und wählen Sie Erlaubt API Gateway, Logs in Logs zu CloudWatch übertragen.
5. Klicken Sie auf Weiter und dann erneut auf Weiter.

6. Geben Sie für Role name (Rollenname) den Namen **APIGatewayKinesisProxyPolicy** ein und klicken Sie auf Create role (Rolle erstellen).
7. Wählen Sie in der Liste Roles die Rolle aus, die Sie gerade erstellt haben. Möglicherweise müssen Sie scrollen oder die Rolle über die Suchleiste finden.
8. Wählen Sie für die ausgewählte Rolle die Registerkarte Berechtigungen hinzufügen aus.
9. Wählen Sie in der Dropdown-Liste Berechtigungen anfügen aus.
10. Geben Sie im Suchfeld **AmazonKinesisFullAccess** ein und wählen Sie Berechtigungen hinzufügen aus.

 Note

Dieses Tutorial verwendet der Einfachheit halber eine verwaltete Richtlinie. Als Best Practice sollten Sie Ihre eigene IAM-Richtlinie erstellen, um die erforderlichen Mindestberechtigungen zu gewähren.

11. Notieren Sie sich den neu erstellten Rollen-ARN, Sie werden ihn später brauchen.

Eine API als Kinesis-Proxy erstellen

Führen Sie die folgenden Schritte aus, um die API in der API Gateway-Konsole zu erstellen.

Um eine API als AWS Service-Proxy für Kinesis zu erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster Create Example API (Beispiel-API erstellen) angezeigt wird, klicken Sie auf OK.

Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.

3. Klicken Sie auf New API (Neue API).
4. Geben Sie unter API name (API-Name) **KinesisProxy** ein. Belassen Sie alle Standardwerte für die anderen Felder unverändert.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.

6. Wählen Sie als IP-Adresstyp die Option. IPv4
7. Wählen Sie Create API (API erstellen) aus.

Nachdem die API erstellt wurde, zeigt die API Gateway-Konsole die Seite Resources (Ressourcen) an, die nur die Root-Ressource (/) der API enthält.

Streams in Kinesis auflisten

Kinesis unterstützt die Aktion `ListStreams` mit dem folgenden REST-API-Aufruf:

```
POST /?Action=ListStreams HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>

{
  ...
}
```

In der oben gezeigten REST-API-Anforderung wird die Aktion im `Action`-Abfrageparameter angegeben. Alternativ können Sie stattdessen die Aktion in einer `X-Amz-Target`-Kopfzeile angeben:

```
POST / HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>
X-Amz-Target: Kinesis_20131202.ListStreams

{
  ...
}
```

In diesem Tutorial verwenden wir den Abfrageparameter, um Aktionen festzulegen.

Um eine Kinesis-Aktion in der API bereitzustellen, fügen Sie eine `/streams`-Ressource zum Stamm der API hinzu. Richten Sie dann eine GET-Methode für die Ressource ein und integrieren Sie die Methode mit der `ListStreams`-Aktion von Kinesis.

Das folgende Verfahren beschreibt die Auflistung von Kinesis-Streams mit Hilfe der API Gateway-Konsole.

So listen Sie Kinesis-Streams über die API Gateway-Konsole auf:

1. Wählen Sie die `/`-Ressource aus und klicken Sie dann auf Ressource erstellen.
2. Geben Sie für Resource name (Ressourcenname) **streams** ein.
3. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
4. Wählen Sie Create Resource (Ressource erstellen) aus.
5. Wählen Sie die `/streams`-Ressource aus, klicken Sie auf Methode erstellen und gehen Sie dann wie folgt vor:
 - a. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.

 Note

Das HTTP-Verb für eine Methode, die von einem Client aufgerufen wird, kann sich von dem HTTP-Verb unterscheiden, das für eine Integration für das Backend erforderlich ist. Hier wählen wir GET aus, da das Auflisten von Streams intuitiv ein READ-Vorgang ist.

- b. Als Integrationstyp wählen Sie AWS -Dienst aus.
- c. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
- d. Für AWS-Service wählen Sie Kinesis aus.
- e. Lassen Sie die AWS -Subdomain leer.
- f. Wählen Sie in HTTP method POST.

 Note

Wir haben hier POST gewählt, weil Kinesis erfordert, dass die Aktion `ListStreams` mit aufgerufen wird.

- g. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
 - h. Für Aktionsname geben Sie **ListStreams** ein.
 - i. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
 - j. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
 - k. Wählen Sie Methode erstellen aus.
6. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
7. Wählen Sie für Anforderungstext-Pass-Through die Einstellung Wenn keine Vorlagen definiert sind (empfohlen) aus.
8. Klicken Sie auf URL-Abfragezeichenfolgen-Parameter und gehen Sie dann wie folgt vor:
- a. Klicken Sie auf Parameter Anforderungsheader hinzufügen.
 - b. Geben Sie unter Name **Content-Type** ein.
 - c. Geben Sie für Zugeordnet von **'application/x-amz-json-1.1'** ein.

Wir verwenden eine Zuweisung für den Anfrageparameter, um den Content-Type-Header auf den statischen Wert **'application/x-amz-json-1.1'** einzustellen und damit Kinesis mitzuteilen, dass die Eingabe in einer bestimmten JSON-Version erfolgt.

9. Wählen Sie Mapping-Vorlagen und dann Mapping-Vorlage hinzufügen aus und gehen Sie dann wie folgt vor:
- a. Geben Sie für Inhaltstyp **application/json** ein.
 - b. Geben Sie für Vorlage **{}** ein.
 - c. Wählen Sie Speichern.

Die [ListStreams](#)Anfrage benötigt eine Nutzlast im folgenden JSON-Format:

```
{
  "ExclusiveStartStreamName": "string",
  "Limit": number
}
```

Die Eigenschaften sind jedoch optional. Wir haben uns für eine leere JSON-Nutzlast entschieden, um die Standardwerte zu verwenden.

10. Testen Sie die GET-Methode für die /streams-Ressource zum Aufrufen der ListStreams-Aktion in Kinesis:

Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

Klicken Sie auf Test, um Ihre Methode zu testen.

Wenn Sie bereits zwei Streams mit den Namen „myStream“ und „yourStream“ in Kinesis erstellt haben, gibt der Test im Erfolgsfall die Antwort „200 OK“ mit der folgenden Nutzlast zurück:

```
{
  "HasMoreStreams": false,
  "StreamNames": [
    "myStream",
    "yourStream"
  ]
}
```

Stream in Kinesis erstellen, beschreiben und löschen

Das Erstellen, Beschreiben und Löschen eines Streams in Kinesis umfasst jeweils die folgenden Kinesis REST-API-Anfragen:

```
POST /?Action=CreateStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "ShardCount": number,
  "StreamName": "string"
}
```

```
POST /?Action=DescribeStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "StreamName": "string"
}
```

```
POST /?Action>DeleteStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "StreamName": "string"
}
```

Sie können die API so erstellen, dass sie die erforderliche Eingabe als JSON-Nutzlast der Methodenanforderung akzeptiert und die Nutzlast bis zur Integrationsanforderung weiterleitet. Die API wird jedoch geringfügig abgewandelt, um weitere Beispiele für die Datenzuweisung zwischen Methoden- und Integrationsanforderungen sowie Methoden- und Integrationsantworten bereitzustellen.

Wir machen die MethodenGET,POST, und Delete HTTP für eine to-be-named Stream Ressource verfügbar. Sie verwenden die Pfadvariable {stream-name} als Platzhalter für die Stream-Ressource und integrieren diese API-Methoden in die entsprechenden DescribeStream-, CreateStream- und DeleteStream-Aktionen von Kinesis. Wir benötigen, dass der Client andere Eingabedaten als Header, Abfrageparameter oder Nutzdaten einer Methodenanforderung weitergibt. Wir stellen Mapping-Vorlagen zur Verfügung, um die Daten in die erforderliche Nutzlast der Integrationsanforderung zu transformieren.

Erstellen einer {Stream-name}-Ressource

1. Wählen Sie die Ressource `/streams` aus und klicken Sie dann auf Ressource erstellen.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Wählen Sie für Ressourcenpfad `/streams` aus.
4. Geben Sie für Resource name (Ressourcenname) **{stream-name}** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

So konfigurieren und testen Sie die GET-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource `{stream-name}` aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **DescribeStream** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

```
Content-Type: 'x-amz-json-1.1'
```

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode GET `/streams` einzurichten.

14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung GET /streams/{stream-name} der Integrationsanforderung POST /? Action=DescribeStream zuzuweisen:

```
{
  "StreamName": "$input.params('stream-name')"
}
```

Diese Zuweisungsvorlage generiert die erforderliche Nutzlast der Integrationsanforderung für die Aktion DescribeStream von Kinesis aufgrund des Pfadparameterwerts stream-name der Methodenanforderung.

15. Klicken Sie auf die Registerkarte Test, um die GET /stream/{stream-name}-Methode zum Aufrufen der DescribeStream-Aktion in Kinesis zu testen.
16. Geben Sie unter Stream-Name den Namen eines vorhandenen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" mit einer Nutzlast ähnlich der folgenden zurückgegeben:

```
{
  "StreamDescription": {
    "HasMoreShards": false,
    "RetentionPeriodHours": 24,
    "Shards": [
      {
        "HashKeyRange": {
          "EndingHashKey": "68056473384187692692674921486353642290",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
"49559266461454070523309915164834022007924120923395850242"
        },
        "ShardId": "shardId-000000000000"
      },
      ...
      {
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768211455",
          "StartingHashKey": "272225893536750770770699685945414569164"
        },
        "SequenceNumberRange": {
```

```
    "StartingSequenceNumber":  
      "49559266461543273504104037657400164881014714369419771970"  
    },  
    "ShardId": "shardId-000000000004"  
  }  
],  
  "StreamARN": "arn:aws:kinesis:us-east-1:12345678901:stream/myStream",  
  "StreamName": "myStream",  
  "StreamStatus": "ACTIVE"  
}  
}
```

Nach der Bereitstellung der API können Sie eine REST-Anforderung für diese API-Methode erstellen:

```
GET https://your-api-id.execute-api.region.amazonaws.com/stage/streams/myStream  
HTTP/1.1  
Host: your-api-id.execute-api.region.amazonaws.com  
Content-Type: application/json  
Authorization: ...  
X-Amz-Date: 20160323T194451Z
```

So konfigurieren und testen Sie die POST-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource `/stream-name` aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie unter Method type (Methodentyp) die Option POST aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **CreateStream** ein.

10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

```
Content-Type: 'x-amz-json-1.1'
```

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode GET /streams einzurichten.

14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung POST /streams/{stream-name} der Integrationsanforderung POST /?Action=CreateStream zuzuweisen:

```
{
  "ShardCount": #if($input.path('$.ShardCount') == '') 5 #else
    $input.path('$.ShardCount') #end,
  "StreamName": "$input.params('stream-name')"
}
```

In der vorhergehenden Zuweisungsvorlage legen wir ShardCount auf einen festen Wert von 5 fest, wenn vom Client kein Wert in der Nutzlast der Methodenanforderung angegeben wird.

15. Klicken Sie auf die Registerkarte Test, um die POST /stream/{stream-name}-Methode zum Aufrufen der CreateStream-Aktion in Kinesis zu testen.
16. Geben Sie für Path unter Stream-Name den Namen eines neuen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" ohne Daten zurückgegeben.

Nach der Bereitstellung der API können Sie auch eine REST-API-Anfrage im Hinblick auf die POST-Methode für eine Stream-Ressource erstellen, um die CreateStream-Aktion in Amazon Kinesis aufzurufen:

```
POST https://your-api-id.execute-api.region.amazonaws.com/stage/streams/yourStream
HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
```

```
Authorization: ...
X-Amz-Date: 20160323T194451Z

{
  "ShardCount": 5
}
```

Konfigurieren und Testen der DELETE-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource `{stream-name}` aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp LÖSCHEN aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **DeleteStream** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

```
Content-Type: 'x-amz-json-1.1'
```

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode GET `/streams` einzurichten.

14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung DELETE `/streams/{stream-name}` der entsprechenden Integrationsanforderung POST `/?Action=DeleteStream` zuzuweisen:

```
{
```

```
"StreamName": "$input.params('stream-name')"
```

```
}
```

Diese Zuweisungsvorlage generiert die erforderliche Eingabe für die Aktion DELETE / streams/{stream-name} aus dem vom Client bereitgestellten URL-Pfad stream-name.

15. Klicken Sie auf die Registerkarte Test, um die DELETE /stream/{stream-name}-Methode zum Aufrufen der DeleteStream-Aktion in Kinesis zu testen.
16. Geben Sie unter Stream-Name den Namen eines vorhandenen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" ohne Daten zurückgegeben.

Nach der Bereitstellung der API können Sie auch die folgende REST-API-Anfrage im Hinblick auf die DELETE-Methode für die Stream-Ressource erstellen, um die DeleteStream-Aktion in Amazon Kinesis aufzurufen:

```
DELETE https://your-api-id.execute-api.region.amazonaws.com/stage/
streams/yourStream HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{}
```

Datensätze aus einem Stream in Kinesis abrufen und Datensätze zu einem Stream hinzufügen

Nachdem Sie einen Stream in Kinesis erstellt haben, können Sie dem Stream Datensätze hinzufügen und die Daten aus dem Stream lesen. Zum Hinzufügen von Datensätzen muss die [PutRecord](#)Aktion [PutRecords](#)oder in Kinesis aufgerufen werden. Erstere fügt dem Stream mehrere Datensätze hinzu, letztere hingegen einen einzigen Datensatz.

```
POST /?Action=PutRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
```

```
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "Records": [
    {
      "Data": blob,
      "ExplicitHashKey": "string",
      "PartitionKey": "string"
    }
  ],
  "StreamName": "string"
}
```

oder

```
POST /?Action=PutRecord HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "Data": blob,
  "ExplicitHashKey": "string",
  "PartitionKey": "string",
  "SequenceNumberForOrdering": "string",
  "StreamName": "string"
}
```

Hier identifiziert `StreamName` den Zieldatenstrom zum Hinzufügen von Datensätzen. `StreamName`, `Data`, und `PartitionKey` sind erforderliche Eingabedaten. In diesem Beispiel werden die Standardwerte für alle optionalen Eingabedaten verwendet. Es erfolgt keine explizite Angabe entsprechender Werte in der Eingabe für die Methodenanforderung.

Das Lesen von Daten in Kinesis läuft darauf hinaus, die [GetRecords](#)Aktion aufzurufen:

```
POST /?Action=GetRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
  "ShardIterator": "string",
  "Limit": number
}
```

Hier wird der Quell-Stream, von dem die Datensätze abgerufen werden, im erforderlichen `ShardIterator`-Wert angegeben. Dies wird in der folgenden Kinesis-Aktion zum Abrufen eines gemeinsamen Iterators verdeutlicht:

```
POST /?Action=GetShardIterator HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

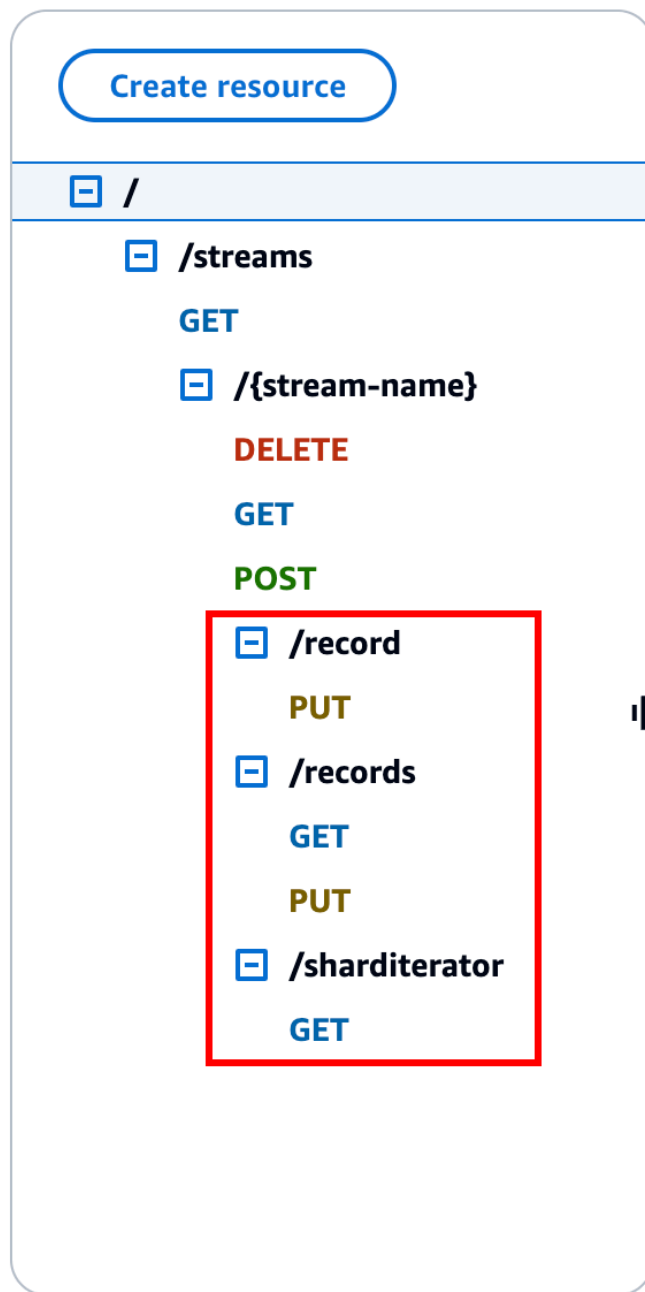
{
  "ShardId": "string",
  "ShardIteratorType": "string",
  "StartingSequenceNumber": "string",
  "StreamName": "string"
}
```

Für die Aktionen `GetRecords` und `PutRecords` werden die Methoden `GET` und `PUT` entsprechend für eine `/records`-Ressource bereitgestellt, die einem benannten Stream hinzugefügt wird (`/ {stream-name}`). Ebenso wird die `PutRecord`-Aktion als `PUT`-Methode für eine `/record`-Ressource verfügbar gemacht.

Da die Aktion `GetRecords` als Eingabe einen `ShardIterator`-Wert verwendet, der durch Aufrufen der Hilfsaktion `GetShardIterator` gewonnen wurde, stellen Sie eine `GET`-Hilfsmethode für eine `ShardIterator`-Ressource bereit (`/sharditerator`).

So erstellen Sie die Ressourcen `/record`, `/records` und `/sharditerator`

1. Wählen Sie die Ressource `/stream-name` aus und klicken Sie dann auf Ressource erstellen.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Wählen Sie für Ressourcenpfad `/stream-name` aus.
4. Geben Sie für Resource name (Ressourcenname) **record** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.
7. Wiederholen Sie die Schritte oben, um eine `/records`- und eine `/sharditerator`-Ressource zu erstellen. Die endgültige API sollte wie folgt aussehen:



In den folgenden vier Verfahren wird beschrieben, wie Sie die jeweiligen Methoden einrichten, Daten aus den Methodenanforderungen den Integrationsanforderungen zuweisen und die Methoden testen.

Einrichten und Testen der **PUT /streams/{stream-name}/record**-Methode zum Aufruf von **PutRecord** in Kinesis:

1. Wählen Sie **/record** aus und klicken Sie dann auf **Methode erstellen**.

2. Wählen Sie unter Methodentyp PUT aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **PutRecord** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

```
Content-Type: 'x-amz-json-1.1'
```

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode GET /streams einzurichten.

14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung PUT /streams/{stream-name}/record der entsprechenden Integrationsanforderung POST /?Action=PutRecord zuzuweisen:

```
{
  "StreamName": "$input.params('stream-name')",
  "Data": "$util.base64Encode($input.json('$.Data'))",
  "PartitionKey": "$input.path('$.PartitionKey')"
}
```

In dieser Zuweisungsvorlage wird davon ausgegangen, dass die Nutzlast der Methodenanforderung folgendes Format aufweist:

```
{
  "Data": "some data",
```

```
"PartitionKey": "some key"
}
```

Diese Daten können mithilfe des folgenden JSON-Schemas modelliert werden:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecord proxy single-record payload",
  "type": "object",
  "properties": {
    "Data": { "type": "string" },
    "PartitionKey": { "type": "string" }
  }
}
```

Sie können ein Modell mit diesem Schema erstellen und es verwenden, um die Generierung der Zuweisungsvorlage zu vereinfachen. Sie können eine Zuweisungsvorlage jedoch auch ohne ein Modell erstellen.

15. Testen Sie die Methode **PUT /streams/{stream-name}/record**, indem Sie die Pfadvariable `stream-name` auf den Namen eines vorhandenen Streams festlegen, eine Nutzlast im erforderlichen Format bereitstellen und die Methodenanforderung dann senden. Das Ergebnis im Erfolgsfall ist die Antwort **200 OK** mit einer Nutzlast in folgendem Format:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-0000000000004"
}
```

So richten Sie die **PUT /streams/{stream-name}/records**-Methode zum Aufruf von **PutRecords** in Kinesis ein und testen sie:

1. Wählen Sie die Ressource `/records` aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie unter Methodentyp **PUT** aus.
3. Als Integrationstyp wählen Sie **AWS -Dienst** aus.
4. Wählen Sie für den AWS-Region Ort aus **AWS-Region**, an dem Sie Ihren Kinesis-Stream erstellt haben.

5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **PutRecords** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

```
Content-Type: 'x-amz-json-1.1'
```

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode GET /streams einzurichten.

14. Fügen Sie die folgende Zuweisungsvorlage hinzu, um Daten der PUT /streams/{stream-name}/records-Methodenanforderung der entsprechenden POST /?Action=PutRecords-Integrationsanforderung zuzuweisen:

```
{
  "StreamName": "$input.params('stream-name')",
  "Records": [
    #foreach($elem in $input.path('$.records'))
    {
      "Data": "$util.base64Encode($elem.data)",
      "PartitionKey": "$elem.partition-key"
    }#if($foreach.hasNext),#end
  ]#end
}
```

In dieser Zuweisungsvorlage wird davon ausgegangen, dass die Nutzlast der Methodenanforderung anhand des folgenden JSON-Schemas modelliert werden kann:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecords proxy payload data",
```

```
"type": "object",
"properties": {
  "records": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "data": { "type": "string" },
        "partition-key": { "type": "string" }
      }
    }
  }
}
```

Sie können ein Modell mit diesem Schema erstellen und es verwenden, um die Generierung der Zuweisungsvorlage zu vereinfachen. Sie können eine Zuweisungsvorlage jedoch auch ohne ein Modell erstellen.

In diesem Tutorial haben wir zwei geringfügig unterschiedliche Nutzdatenformate verwendet, um zu veranschaulichen, dass ein API-Entwickler je nach Bedarf das Backend-Datenformat für den Client offenlegen oder gegenüber dem Client verbergen kann. Eines dieser Formate ist für die Methode PUT `/streams/{stream-name}/records` bestimmt (oben). Ein anderes Format wird für die Methode PUT `/streams/{stream-name}/record` (im vorherigen Verfahren) verwendet. In einer Produktionsumgebung sollten beide Formate konsistent bleiben.

15. Testen Sie die Methode PUT `/streams/{stream-name}/records`, indem Sie die Pfadvariable `stream-name` auf einen vorhandenen Stream festlegen, die folgende Nutzlast bereitstellen und die Methodenanforderung senden.

```
{
  "records": [
    {
      "data": "some data",
      "partition-key": "some key"
    },
    {
      "data": "some other data",
      "partition-key": "some key"
    }
  ]
}
```

```
]
}
```

Das Ergebnis im Erfolgsfall ist die Antwort "200 OK" mit einer Nutzlast, die der folgenden Ausgabe ähnelt:

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber": "49559409944537880850133345460167468741933742152373764162",
      "ShardId": "shardId-000000000004"
    },
    {
      "SequenceNumber": "49559409944537880850133345460168677667753356781548470338",
      "ShardId": "shardId-000000000004"
    }
  ]
}
```

Richten Sie die **GET /streams/{stream-name}/sharditerator**-Methode ein und testen Sie sie, indem Sie **GetShardIterator** in Kinesis aufrufen

Die Methode **GET /streams/{stream-name}/sharditerator** ist eine Hilfsmethode, um einen erforderlichen gemeinsamen Iterator vor dem Aufrufen der Methode **GET /streams/{stream-name}/records** zu beschaffen.

1. Wählen Sie die Ressource **/sharditerator** aus und klicken Sie dann auf **Methode erstellen**.
2. Wählen Sie als Methodentyp die Option **GET (Abrufen)** aus.
3. Als Integrationstyp wählen Sie **AWS -Dienst** aus.
4. Wählen Sie für den AWS-Region Ort aus **AWS-Region**, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie **Kinesis** aus.
6. Lassen Sie die **AWS -Subdomain** leer.
7. Wählen Sie in **HTTP method** **POST**.
8. Für Aktionstyp wählen Sie **Aktionsname verwenden** aus.

9. Für Aktionsname geben Sie **GetShardIterator** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

Die `GetShardIterator`-Aktion erfordert die Eingabe eines `ShardId`-Werts. Zur Weitergabe eines vom Client bereitgestellten `ShardId`-Werts wird der Methodenanforderung ein `shard-id`-Abfrageparameter wie nachfolgend dargestellt hinzugefügt:

13. Wählen Sie Abfragezeichenfolge hinzufügen aus.
14. Geben Sie unter Name **shard-id** ein.
15. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
16. Wählen Sie Methode erstellen aus.
17. Fügen Sie im Abschnitt Integrationsanforderung die folgende Zuweisungsvorlage hinzu, um die erforderlichen Eingaben (`ShardId` und `StreamName`) für die `GetShardIterator`-Aktion aus den Parametern `shard-id` und `stream-name` der Methodenanforderung zu generieren. Durch die Zuweisungsvorlage wird außerdem `ShardIteratorType` auf die Standardeinstellung `TRIM_HORIZON` festgelegt.

```
{
  "ShardId": "$input.params('shard-id')",
  "ShardIteratorType": "TRIM_HORIZON",
  "StreamName": "$input.params('stream-name')"
}
```

18. Verwenden Sie die Option Test in der Amazon API Gateway-Konsole, geben Sie einen vorhandenen Stream-Namen als Wert der Variablen `stream-name` Path ein, setzen Sie die `shard-id` Query string auf einen vorhandenen `ShardId`-Wert (z. B. `shard-0000000000004`) und wählen Sie Test.

Im Erfolgsfall sieht die Ausgabe der Antwortnutzlast folgendermaßen oder ähnlich aus:

```
{
  "ShardIterator": "AAAAAAAAAFYVN3VlFy..."
}
```

Notieren Sie sich den Wert für `ShardIterator`. Sie benötigen ihn, um Datensätze aus einem Stream abzurufen.

So konfigurieren und testen Sie die **GET /streams/{stream-name}/records**-Methode zum Aufrufen der **GetRecords**-Aktion in Kinesis:

1. Wählen Sie die Ressource `/records` aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **GetRecords** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie HTTP-Anforderungs-Headers aus.

Die GetRecords-Aktion erfordert die Eingabe eines `ShardIterator`-Werts. Für eine Weitergabe eines vom Client bereitgestellten `ShardIterator`-Werts muss der Methodenanforderung ein `Shard-Iterator-Header-Parameter` hinzugefügt werden.

13. Wählen Sie Add header.
14. Geben Sie unter Name **Shard-Iterator** ein.
15. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
16. Wählen Sie Methode erstellen aus.
17. Fügen Sie die folgende Textzuweisungsvorlage im Abschnitt Integrationsanforderung hinzu, um den `Shard-Iterator-Header-Parameter`wert dem `ShardIterator`-Eigenschaftswert der JSON-Nutzlast für die GetRecords-Aktion in Kinesis zuzuweisen.

```
{
  "ShardIterator": "$input.params('Shard-Iterator')"
}
```

18. Mithilfe der Option Test in der Amazon-API-Gateway-Konsole geben Sie einen vorhandenen Stream-Namen als Wert der Variablen `stream-name-Pfad` ein, setzen den `Shard-`

Iterator-Header auf den ShardIterator-Wert aus dem Testdurchlauf der GET `/streams/{stream-name}/sharditerator`-Methode (oben) und klicken dann auf Test.

Im Erfolgsfall sieht die Ausgabe der Antwortnutzlast folgendermaßen oder ähnlich aus:

```
{
  "MillisBehindLatest": 0,
  "NextShardIterator": "AAAAAAAAAAAF...",
  "Records": [ ... ]
}
```

OpenAPI-Definitionen einer Beispiel-API als Kinesis-Proxy

Nachfolgend finden Sie OpenAPI-Definitionen für die in diesem Tutorial verwendete Beispiel-API als Kinesis-Proxy.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "KinesisProxy",
    "version": "2016-03-31T18:25:32Z"
  },
  "paths": {
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "shard-id",
            "in": "query",
            "schema": {
              "type": "string"
            }
          }
        ]
      }
    }
  }
}
```

```

    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/Empty"
          }
        }
      }
    }
  }
},
"x-amazon-apigateway-integration": {
  "type": "aws",
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
  "responses": {
    "default": {
      "statusCode": "200"
    }
  },
  "requestParameters": {
    "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
  },
  "requestTemplates": {
    "application/json": "{\n  \"ShardId\": \"$input.params('shard-
id')\",\n  \"ShardIteratorType\": \"TRIM_HORIZON\",\n  \"StreamName\":
  \"$input.params('stream-name')\"\n}"
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "POST"
}
},
"/streams/{stream-name}/records": {
  "get": {
    "parameters": [
      {
        "name": "stream-name",
        "in": "path",
        "required": true,

```

```

        "schema": {
            "type": "string"
        }
    },
    {
        "name": "Shard-Iterator",
        "in": "header",
        "schema": {
            "type": "string"
        }
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "content": {
            "application/json": {
                "schema": {
                    "$ref": "#/components/schemas/Empty"
                }
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestParameters": {
        "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
        "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
}

```

```
    },
    "put": {
      "parameters": [
        {
          "name": "Content-Type",
          "in": "header",
          "schema": {
            "type": "string"
          }
        },
        {
          "name": "stream-name",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
          },
          "application/x-amz-json-1.1": {
            "schema": {
              "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
          }
        },
        "required": true
      },
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
              }
            }
          }
        }
      }
    }
  }
```

```

    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
    "responses": {
      "default": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n  \"StreamName\": \"$input.params('stream-
name')\",\n  \"Records\": [\n    {\n      \"Data\":\n        \"${util.base64Encode($elem.data)}\",\n      \"PartitionKey\":\n        \"${elem.partition-key}\"#if($foreach.hasNext),#end\n    ]\n}",
      "application/x-amz-json-1.1": "{\n  \"StreamName\":\n    \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n      \"Data\
\n\" : \"${elem.data}\",\n      \"PartitionKey\" : \"${elem.partition-key}\"#if($foreach.hasNext),#end\n    ]\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
}
},
"/streams/{stream-name}": {
  "get": {
    "parameters": [
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "schema": {
          "type": "string"
        }
      }
    ],
    "responses": {
      "200": {

```

```

        "description": "200 response",
        "content": {
            "application/json": {
                "schema": {
                    "$ref": "#/components/schemas/Empty"
                }
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"${input.params('stream-
name')}\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"post": {
    "parameters": [
        {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
                "type": "string"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "content": {
                "application/json": {
                    "schema": {

```

```

        "$ref": "#/components/schemas/Empty"
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
    "responses": {
      "default": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
\n$input.params('stream-name')\n\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"delete": {
  "parameters": [
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "schema": {
        "type": "string"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "headers": {
        "Content-Type": {
          "schema": {

```



```

        "type": "string"
      }
    }
  },
  "content": {
    "application/json": {
      "schema": {
        "$ref": "#/components/schemas/Empty"
      }
    }
  }
},
"400": {
  "description": "400 response",
  "headers": {
    "Content-Type": {
      "schema": {
        "type": "string"
      }
    }
  },
  "content": {}
},
"500": {
  "description": "500 response",
  "headers": {
    "Content-Type": {
      "schema": {
        "type": "string"
      }
    }
  },
  "content": {}
}
},
"x-amazon-apigateway-integration": {
  "type": "aws",
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
  "responses": {
    "4\\d{2}": {
      "statusCode": "400",
      "responseParameters": {

```

```

        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
    },
    "default": {
        "statusCode": "200",
        "responseParameters": {
            "method.response.header.Content-Type":
"integration.response.header.Content-Type"
        },
        "5\\d{2}": {
            "statusCode": "500",
            "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
            }
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"${input.params('stream-
name')}\n\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"/streams/{stream-name}/record": {
    "put": {
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "schema": {
                    "type": "string"
                }
            }
        ]
    },

```

```

    "responses": {
      "200": {
        "description": "200 response",
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Empty"
            }
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "type": "aws",
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
      "responses": {
        "default": {
          "statusCode": "200"
        }
      },
      "requestParameters": {
        "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
      },
      "requestTemplates": {
        "application/json": "{\n  \"StreamName\": \"${input.params('stream-
name')}\",\n  \"Data\": \"${util.base64Encode($input.json('$.Data'))}\",\n
  \"PartitionKey\": \"${input.path('$.PartitionKey')}\",\n}"
      },
      "passthroughBehavior": "when_no_match",
      "httpMethod": "POST"
    }
  },
  "/streams": {
    "get": {
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
              }
            }
          }
        }
      }
    }
  }
}

```

```

        }
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
    "responses": {
      "default": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
}
},
"components": {
  "schemas": {
    "Empty": {
      "type": "object"
    },
    "PutRecordsMethodRequestPayload": {
      "type": "object",
      "properties": {
        "records": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "data": {
                "type": "string"
              },
              "partition-key": {

```

```

        "type": "string"
      }
    }
  }
}

```

OpenAPI 2.0

```

{
  "swagger": "2.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/streams": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",

```

```

        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"/streams/{stream-name}": {
    "get": {
        "consumes": [
            "application/json"
        ],
        "produces": [
            "application/json"
        ],
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "type": "string"
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "schema": {
                    "$ref": "#/definitions/Empty"
                }
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",

```

```

        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"post": {
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "parameters": [
        {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "schema": {
                "$ref": "#/definitions/Empty"
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
    "responses": {
        "default": {

```

```

        "statusCode": "200"
      },
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
\\\"$input.params('stream-name')\\\"\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"delete": {
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "type": "string"
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Empty"
      },
      "headers": {
        "Content-Type": {
          "type": "string"
        }
      }
    },
    "400": {
      "description": "400 response",

```



```

        "headers": {
            "Content-Type": {
                "type": "string"
            }
        },
        "500": {
            "description": "500 response",
            "headers": {
                "Content-Type": {
                    "type": "string"
                }
            }
        },
        "x-amazon-apigateway-integration": {
            "type": "aws",
            "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
            "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
            "responses": {
                "4\\d{2}": {
                    "statusCode": "400",
                    "responseParameters": {
                        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                    }
                },
                "default": {
                    "statusCode": "200",
                    "responseParameters": {
                        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                    }
                },
                "5\\d{2}": {
                    "statusCode": "500",
                    "responseParameters": {
                        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                    }
                }
            },
            "requestParameters": {

```

```

        "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
      },
      "requestTemplates": {
        "application/json": "{\n  \n  \"StreamName\": \"\${input.params('stream-
name')}\n\n}"
      },
      "passthroughBehavior": "when_no_match",
      "httpMethod": "POST"
    }
  },
  "/streams/{stream-name}/record": {
    "put": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "stream-name",
          "in": "path",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "type": "aws",
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
      "responses": {
        "default": {
          "statusCode": "200"
        }
      }
    }
  }
}

```

```

    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n  \"StreamName\": \"${input.params('stream-
name')}\",\n  \"Data\": \"${util.base64Encode($input.json('$.Data'))}\",\n
  \"PartitionKey\": \"${input.path('$.PartitionKey')}\",\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"/streams/{stream-name}/records": {
  "get": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "type": "string"
      },
      {
        "name": "Shard-Iterator",
        "in": "header",
        "required": false,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "200 response",
        "schema": {
          "$ref": "#/definitions/Empty"
        }
      }
    }
  }
}

```

```

    },
    "x-amazon-apigateway-integration": {
      "type": "aws",
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
      "responses": {
        "default": {
          "statusCode": "200"
        }
      },
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"put": {
  "consumes": [
    "application/json",
    "application/x-amz-json-1.1"
  ],
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "name": "Content-Type",
      "in": "header",
      "required": false,
      "type": "string"
    },
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "type": "string"
    }
  ],
  {

```

```

        "in": "body",
        "name": "PutRecordsMethodRequestPayload",
        "required": true,
        "schema": {
            "$ref": "#/definitions/PutRecordsMethodRequestPayload"
        }
    },
    {
        "in": "body",
        "name": "PutRecordsMethodRequestPayload",
        "required": true,
        "schema": {
            "$ref": "#/definitions/PutRecordsMethodRequestPayload"
        }
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "schema": {
            "$ref": "#/definitions/Empty"
        }
    }
},
"x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestParameters": {
        "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
        "application/json": "{\n    \"StreamName\": \"${input.params('stream-
name')}\",\n    \"Records\": [\n        {\n            \"Data\":\n                \"${util.base64Encode($elem.data)}\", \n            \"PartitionKey\":\n                \"${elem.partition-key}\",\n        }#if($foreach.hasNext),#end\n    ]\n}",
        "application/x-amz-json-1.1": "{\n    \"StreamName\":\n        \"${input.params('stream-name')}\",\n    \"records\" : [\n        {\n            \"Data

```

```

\" : \"$elem.data\", \n      \"PartitionKey\" : \"$elem.partition-key\" \n
}#if($foreach.hasNext),#end \n  ] \n\"
    },
    \"passthroughBehavior\": \"when_no_match\",
    \"httpMethod\": \"POST\"
  }
}
},
\"/streams/{stream-name}/sharditerator\": {
  \"get\": {
    \"consumes\": [
      \"application/json\"
    ],
    \"produces\": [
      \"application/json\"
    ],
    \"parameters\": [
      {
        \"name\": \"stream-name\",
        \"in\": \"path\",
        \"required\": true,
        \"type\": \"string\"
      },
      {
        \"name\": \"shard-id\",
        \"in\": \"query\",
        \"required\": false,
        \"type\": \"string\"
      }
    ],
    \"responses\": {
      \"200\": {
        \"description\": \"200 response\",
        \"schema\": {
          \"$ref\": \"#/definitions/Empty\"
        }
      }
    }
  },
  \"x-amazon-apigateway-integration\": {
    \"type\": \"aws\",
    \"credentials\": \"arn:aws:iam::123456789012:role/apigAwsProxyRole\",
    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator\",
    \"responses\": {
      \"default\": {

```

```

        "statusCode": "200"
      },
      {
        "requestParameters": {
          "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
          "application/json": "{\n  \n  \"ShardId\": \"${input.params('shard-
id')}\",\n  \n  \"ShardIteratorType\": \"TRIM_HORIZON\",\n  \n  \"StreamName\":
\n  \"${input.params('stream-name')}\",\n  \n  }\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
      }
    ]
  },
  {
    "definitions": {
      "Empty": {
        "type": "object"
      },
      "PutRecordsMethodRequestPayload": {
        "type": "object",
        "properties": {
          "records": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "data": {
                  "type": "string"
                },
                "partition-key": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}

```

Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI

Verwenden Sie das folgende Tutorial, um eine PetStore API zu erstellen, die die GET `/pets/{petId}` Methoden GET `/pets` und unterstützt. Die Methoden sind mit einem HTTP-Endpunkt integriert. Sie können diesem Tutorial mit dem AWS SDK für JavaScript, dem SDK for Python (Boto3) oder dem folgen. AWS CLI Das API richten Sie mithilfe der folgenden Funktionen und Befehle ein:

JavaScript v3

- [CreateRestApiCommand](#)
- [CreateResourceCommand](#)
- [PutMethodCommand](#)
- [PutMethodResponseCommand](#)
- [PutIntegrationCommand](#)
- [PutIntegrationResponseCommand](#)
- [CreateDeploymentCommand](#)

Python

- [create_rest_api](#)
- [create_resource](#)
- [put_method](#)
- [put_method_response](#)
- [put_integration](#)
- [put_integration_response](#)
- [create_deployment](#)

AWS CLI

- [create-rest-api](#)
- [create-resource](#)
- [put-method](#)
- [put-method-response](#)
- [put-integration](#)

- [put-integration-response](#)
- [create-deployment](#)

Weitere Informationen zum AWS SDK für JavaScript v3 finden Sie unter [Wofür ist das AWS SDK? JavaScript](#) . Weitere Informationen zum SDK für Python (Boto3) finden Sie unter [AWS SDK für Python \(Boto3\)](#). Weitere Informationen zu dem AWS CLI finden Sie unter [Was ist der AWS CLI?](#) .

Richten Sie eine Edge-optimierte API PetStore ein

In diesem Tutorial verwenden Beispielbefehle Platzhalterwerte für Werte IDs wie API-ID und Ressourcen-ID. Nach Abschluss des Tutorials ersetzen Sie die Beispielwerte mit Ihren eigenen Werten.

Um eine PetStore Edge-optimierte API einzurichten, verwenden Sie AWS SDKs

1. Erstellen Sie mithilfe des folgenden Beispiels eine RestApi-Entity:

JavaScript v3

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
(async function (){
  const apig = new APIGatewayClient({region:"us-east-1"});
  const command = new CreateRestApiCommand({
    name: "Simple PetStore (JavaScript v3 SDK)",
    description: "Demo API created using the AWS SDK for JavaScript v3",
    version: "0.00.001",
    binaryMediaTypes: [
      '*'
    ]
  });
  try {
    const results = await apig.send(command)
    console.log(results)
  } catch (err) {
    console.error(Couldn't create API:\n", err)
  }
})();
```

Ein erfolgreicher Aufruf gibt Ihre API-ID und die Root-Ressourcen-ID Ihrer API in einer Ausgabe wie folgt zurück:

```
{
  id: 'abc1234',
  name: 'PetStore (JavaScript v3 SDK)',
  description: 'Demo API created using the AWS SDK for node.js',
  createdAt: 2017-09-05T19:32:35.000Z,
  version: '0.00.001',
  rootResourceId: 'efg567'
  binaryMediaTypes: [ '*' ]
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_rest_api(
        name='Simple PetStore (Python SDK)',
        description='Demo API created using the AWS SDK for Python',
        version='0.00.001',
        binaryMediaTypes=[
            '*'
        ]
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Couldn't create REST API %s.", error)
    raise

attribute=["id","name","description","createdAt","version","binaryMediaTypes","apiKeyS
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Ihre API-ID und die Root-Ressourcen-ID Ihrer API in einer Ausgabe wie folgt zurück:

```
{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
'Demo API created using the AWS SDK for Python', 'createdAt':
datetime.datetime(2024, 4, 3, 14, 31, 39, tzinfo=tzlocal()), 'version':
```

```
'0.00.001', 'binaryMediaTypes': ['*'], 'apiKeySource': 'HEADER',  
'endpointConfiguration': {'types': ['EDGE']}, 'disableExecuteApiEndpoint':  
False, 'rootResourceId': 'efg567'}
```

AWS CLI

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-  
west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "id": "abcd1234",  
  "name": "Simple PetStore (AWS CLI)",  
  "createdDate": "2022-12-15T08:07:04-08:00",  
  "apiKeySource": "HEADER",  
  "endpointConfiguration": {  
    "types": [  
      "EDGE"  
    ]  
  },  
  "disableExecuteApiEndpoint": false,  
  "rootResourceId": "efg567"  
}
```

Die von Ihnen erstellte API hat die API-ID abcd1234 und die Root-Ressourcen-ID efg567. Diese Werte verwenden Sie nun bei der Einrichtung Ihrer API.

2. Als Nächstes erstellen Sie eine untergeordnete Ressource für die Root-Ressource. Hierzu geben Sie `RootResourceId` als den `parentId`-Eigenschaftswert an. Erstellen Sie anhand des folgenden Beispiels eine `/pets`-Ressource für Ihre API:

JavaScript v3

```
import {APIGatewayClient, CreateResourceCommand} from "@aws-sdk/client-api-  
gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new CreateResourceCommand({  
    restApiId: 'abcd1234',  
    parentId: 'efg567',
```

```
    pathPart: 'pets'
  });
  try {
    const results = await apig.send(command)
    console.log(results)
  } catch (err) {
    console.log("The '/pets' resource setup failed:\n", err)
  }
})();
```

Ein erfolgreicher Aufruf gibt Informationen zur Ressource in einer Ausgabe wie folgt zurück:

```
{
  "path": "/pets",
  "pathPart": "pets",
  "id": "aaa111",
  "parentId": "efg567"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='efg567',
        pathPart='pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Informationen zur Ressource in einer Ausgabe wie folgt zurück:

```
{'id': 'aaa111', 'parentId': 'efg567', 'pathPart': 'pets', 'path': '/pets'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \  
  --region us-west-2 \  
  --parent-id efg567 \  
  --path-part pets
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "id": "aaa111",  
  "parentId": "efg567",  
  "pathPart": "pets",  
  "path": "/pets"  
}
```

Die von Ihnen erstellte /pets-Ressource hat die Ressourcen-ID aaa111. Diesen Wert verwenden Sie bei der Einrichtung Ihrer API.

3. Als Nächstes fügen Sie der /pets-Ressource eine untergeordnete Ressource hinzu. Diese Ressource, /{petId}, besitzt einen Pfadparameter für die {petId}. Wenn Sie einen Teil eines Pfads zu einem Pfadparameter machen wollen, setzen Sie ihn in geschweifte Klammern { }. Erstellen Sie anhand des folgenden Beispiels eine /pets/{petId}-Ressource für Ihre API:

JavaScript v3

```
import {APIGatewayClient, CreateResourceCommand} from "@aws-sdk/client-api-gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new CreateResourceCommand({  
    restApiId: 'abcd1234',  
    parentId: 'aaa111',  
    pathPart: '{petId}'  
  });  
  try {  
    const results = await apig.send(command)
```

```
        console.log(results)
    } catch (err) {
        console.log("The '/pets/{petId}' resource setup failed:\n", err)
    }
  })();
```

Ein erfolgreicher Aufruf gibt Informationen zur Ressource in einer Ausgabe wie folgt zurück:

```
{
  "path": "/pets/{petId}",
  "pathPart": "{petId}",
  "id": "bbb222",
  "parentId": "aaa111"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='aaa111',
        pathPart='{petId}'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets/{petId}' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Informationen zur Ressource in einer Ausgabe wie folgt zurück:

```
{'id': 'bbb222', 'parentId': 'aaa111', 'pathPart': '{petId}', 'path': '/pets/{petId}'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \  
  --region us-west-2 \  
  --parent-id aaa111 \  
  --path-part '{petId}'
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "id": "bbb222",  
  "parentId": "aaa111",  
  "path": "/pets/{petId}",  
  "pathPart": "{petId}"  
}
```

Die von Ihnen erstellte `/pets/{petId}`-Ressource hat die Ressourcen-ID `bbb222`. Diesen Wert verwenden Sie bei der Einrichtung Ihrer API.

4. In den nächsten zwei Schritten fügen Sie HTTP-Methoden zu Ihren Ressourcen hinzu. In diesem Tutorial legen Sie fest, dass die Methoden offenen Zugriff haben sollen. Hierfür setzen Sie `authorization-type` auf `NONE`. Damit nur authentifizierte Benutzer die Methode aufrufen können, können Sie IAM-Rollen und -Richtlinien, einen Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bekannt) oder einen Amazon Cognito-Benutzerpool verwenden. Weitere Informationen finden Sie unter [the section called “Zugriffskontrolle”](#).

Fügen Sie anhand des folgenden Beispiels die `/pets`-HTTP-Methode der GET-Ressource hinzu:

JavaScript v3

```
import {APIGatewayClient, PutMethodCommand} from "@aws-sdk/client-api-gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new PutMethodCommand({  
    restApiId: 'abcd1234',  
    resourceId: 'aaa111',  
    httpMethod: 'GET',  
    authorizationType: 'NONE'  
  });
```

```
try {
  const results = await apig.send(command)
  console.log(results)
} catch (err) {
  console.log("The 'GET /pets' method setup failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE"
}
```

Python

```
import boto3
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        authorizationType='NONE'
    )
except boto3.exceptions.ClientError as error:
    logger.exception("The 'GET /pets' method setup failed: %s", error)
    raise

attribute=["httpMethod","authorizationType","apiKeyRequired"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:


```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \  
  --resource-id aaa111 \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "httpMethod": "GET",  
  "authorizationType": "NONE",  
  "apiKeyRequired": false  
}
```

5. Fügen Sie anhand des folgenden Beispiels die GET-HTTP-Methode zur `/pets/{petId}`-Ressource hinzu und legen Sie die `requestParameters`-Eigenschaft so fest, dass der vom Client bereitgestellte `petId`-Wert an das Backend übergeben wird:

JavaScript v3

```
import {APIGatewayClient, PutMethodCommand} from "@aws-sdk/client-api-gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new PutMethodCommand({  
    restApiId: 'abcd1234',  
    resourceId: 'bbb222',  
    httpMethod: 'GET',  
    authorizationType: 'NONE',  
    requestParameters: {  
      "method.request.path.petId" : true  
    }  
  });  
  try {  
    const results = await apig.send(command)  
    console.log(results)  
  } catch (err) {
```

```
        console.log("The 'GET /pets/{petId}' method setup failed:\n", err)
    }
    })();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "requestParameters": {
    "method.request.path.petId": true
  }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        authorizationType='NONE',
        requestParameters={
            "method.request.path.petId": True
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets/{petId}' method setup failed: %s", error)
    raise

attribute=["httpMethod","authorizationType","apiKeyRequired",
    "requestParameters" ]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False,
  'requestParameters': {'method.request.path.petId': True}}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2 \
  --request-parameters method.request.path.petId=true
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "apiKeyRequired": false,
  "requestParameters": {
    "method.request.path.petId": true
  }
}
```

6. Fügen Sie mithilfe des folgenden Beispiels die 200 OK-Methodenantwort für die GET /pets-Methode hinzu:

JavaScript v3

```
import {APIGatewayClient, PutMethodResponseCommand} from "@aws-sdk/client-api-gateway";
(async function (){
  const apig = new APIGatewayClient({region:"us-east-1"});
  const command = new PutMethodResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    statusCode: '200'
  });
  try {
    const results = await apig.send(command)
```

```
        console.log(results)
    } catch (err) {
        console.log("Set up the 200 OK response for the 'GET /pets' method failed:
\n", err)
    }
    })();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets' method
failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result = {key:result[key] for key in attribute}
logger.info(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \  
  --resource-id aaa111 --http-method GET \  
  --status-code 200 --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "statusCode": "200"  
}
```

7. Fügen Sie mithilfe des folgenden Beispiels die 200 OK-Methodenantwort für die GET `/pets/{petId}`-Methode hinzu:

JavaScript v3

```
import {APIGatewayClient, PutMethodResponseCommand} from "@aws-sdk/client-api-gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new PutMethodResponseCommand({  
    restApiId: 'abcd1234',  
    resourceId: 'bbb222',  
    httpMethod: 'GET',  
    statusCode: '200'  
  });  
  try {  
    const results = await apig.send(command)  
    console.log(results)  
  } catch (err) {  
    console.log("Set up the 200 OK response for the 'GET /pets/{petId}' method failed:\n", err)  
  }  
})();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{  
  "statusCode": "200"  
}
```

```
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets/{petId}'
method failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result = {key:result[key] for key in attribute}
logger.info(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
--resource-id bbb222 --http-method GET \
--status-code 200 --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
  "statusCode": "200"
}
```

8. Konfigurieren Sie anhand des folgenden Beispiels eine Integration für die GET /pets-Methode mit einem HTTP-Endpunkt. Der HTTPS-Endpunkt lautet `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

JavaScript v3

```
import {APIGatewayClient, PutIntegrationCommand } from "@aws-sdk/client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
  restApiId: 'abcd1234',
  resourceId: 'aaa111',
  httpMethod: 'GET',
  type: 'HTTP',
  integrationHttpMethod: 'GET',
  uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
});
try {
  const results = await apig.send(command)
  console.log(results)
} catch (err) {
  console.log("Set up the integration of the 'GET /pets' method of the API failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "httpMethod": "GET",
  "passthroughBehavior": "WHEN_NO_MATCH",
  "cacheKeyParameters": [],
  "type": "HTTP",
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
  "cacheNamespace": "ccc333"
}
```

Python

```
import botocore
import boto3
```

```
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /' method of the API failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
"uri", "cacheNamespace"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-demo-
endpoint.execute-api.com/petstore/pets', 'cacheNamespace': 'ccc333'}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
--resource-id aaa111 --http-method GET --type HTTP \
--integration-http-method GET \
--uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets' \
--region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
  "type": "HTTP",
  "httpMethod": "GET",
```



```

    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "connectionType": "INTERNET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6sxx2j",
    "cacheKeyParameters": []
  }

```

9. Konfigurieren Sie anhand des folgenden Beispiels eine Integration für die GET /pets/{petId}-Methode mit einem HTTP-Endpoint. Der HTTPS-Endpoint lautet `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`. Im folgenden Schritt ordnen Sie den Pfadparameter `petId` dem Integrationsendpunkt-Pfadparameter `id` zu.

JavaScript v3

```

import {APIGatewayClient, PutIntegrationCommand} from "@aws-sdk/client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
  restApiId: 'abcd1234',
  resourceId: 'bbb222',
  httpMethod: 'GET',
  type: 'HTTP',
  integrationHttpMethod: 'GET',
  uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}'
  requestParameters: {
    "integration.request.path.id": "method.request.path.petId"
  }
});
try {
  const results = await apig.send(command)
  console.log(results)
} catch (err) {
  console.log("Set up the integration of the 'GET /pets/{petId}' method of the API failed:\n", err)
}
})();

```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
```

```

    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "cacheNamespace": "ddd444",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    }
}

```

Python

```

import boto3
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='ieps9b05sf',
        resourceId='t8zeb4',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}',
        requestParameters={
            "integration.request.path.id": "method.request.path.petId"
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /pets/{petId}' method of the API failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
    "uri", "cacheNamespace", "requestParameters"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)

```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
  'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-
demo-endpoint.execute-api.com/petstore/pets/{id}', 'cacheNamespace':
  'ddd444', 'requestParameters': {'integration.request.path.id':
  'method.request.path.petId'}}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}' \
  --request-parameters
  '{"integration.request.path.id":"method.request.path.petId"}' \
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
  "type": "HTTP",
  "httpMethod": "GET",
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
  "connectionType": "INTERNET",
  "requestParameters": {
    "integration.request.path.id": "method.request.path.petId"
  },
  "passthroughBehavior": "WHEN_NO_MATCH",
  "timeoutInMillis": 29000,
  "cacheNamespace": "rjkmth",
  "cacheKeyParameters": []
}
```

10. Fügen Sie anhand des folgenden Beispiels die Integrationsantwort der GET /pets-Integration hinzu:

JavaScript v3

```
import {APIGatewayClient, PutIntegrationResponseCommand} from "@aws-sdk/
client-api-gateway";
(async function (){
  const apig = new APIGatewayClient({region:"us-east-1"});
```

```
const command = new PutIntegrationResponseCommand({
  restApiId: 'abcd1234',
  resourceId: 'aaa111',
  httpMethod: 'GET',
  statusCode: '200',
  selectionPattern: ''
});
try {
  const results = await apig.send(command)
  console.log(results)
} catch (err) {
  console.log("The 'GET /pets' method integration response setup failed:\n",
    err)
}
})();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "selectionPattern": "",
  "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
```

```

    logger.exception("Set up the integration response of the 'GET /pets' method
of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)

```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'selectionPattern': '', 'statusCode': '200'}
```

AWS CLI

```

aws apigateway put-integration-response --rest-api-id abcd1234 \
--resource-id aaa111 --http-method GET \
--status-code 200 --selection-pattern "" \
--region us-west-2

```

Die Ausgabe dieses Befehls ist wie folgt:

```

{
  "statusCode": "200",
  "selectionPattern": ""
}

```

11. Fügen Sie anhand des folgenden Beispiels die Integrationsantwort der GET /pets/{petId}-Integration hinzu:

JavaScript v3

```

import {APIGatewayClient, PutIntegrationResponseCommand } from "@aws-sdk/
client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationResponseCommand({
  restApiId: 'abcd1234',
  resourceId: 'bbb222',
  httpMethod: 'GET',
  statusCode: '200',
  selectionPattern: ''
});

```

```
try {
  const results = await apig.send(command)
  console.log(results)
} catch (err) {
  console.log("The 'GET /pets/{petId}' method integration response setup
  failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{
  "selectionPattern": "",
  "statusCode": "200"
}
```

Python

```
import boto3
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets/{petId}'
    method of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Folgendes zurück:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \  
  --resource-id bbb222 --http-method GET  
  --status-code 200 --selection-pattern ""  
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{  
  "statusCode": "200",  
  "selectionPattern": ""  
}
```

Nachdem Sie die Integrationsantwort erstellt haben, kann Ihre API verfügbare Haustiere auf der PetStore Website abfragen und einzelne Haustiere mit einer bestimmten Kennung anzeigen. Sie müssen Ihre fertiggestellte API bereitstellen, damit sie von Ihren Kunden abgerufen werden kann. Wir empfehlen, Ihre API gründlich zu testen, bevor Sie sie bereitstellen.

12. Testen Sie anhand des folgenden Beispiel die GET /pets-Methode:

JavaScript v3

```
import {APIGatewayClient, TestInvokeMethodCommand} from "@aws-sdk/client-api-gateway";  
(async function () {  
  const apig = new APIGatewayClient({region: "us-east-1"});  
  const command = new TestInvokeMethodCommand({  
    restApiId: 'abcd1234',  
    resourceId: 'aaa111',  
    httpMethod: 'GET',  
    pathWithQueryString: '/',  
  });  
  try {  
    const results = await apig.send(command)  
    console.log(results)  
  } catch (err) {  
    console.log("The test on 'GET /pets' method failed:\n", err)  
  }  
})
```

```
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        pathWithQueryString='/',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets' failed: %s", error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
--resource-id aaa111 /
--http-method GET /
--path-with-query-string '/'
```

13. Testen Sie anhand des folgenden Beispiels die GET /pets/{petId}-Methode mit einer petId von 3:

JavaScript v3

```
import {APIGatewayClient, TestInvokeMethodCommand} from "@aws-sdk/client-api-gateway";
(async function (){
    const apig = new APIGatewayClient({region:"us-east-1"});
    const command = new TestInvokeMethodCommand({
        restApiId: 'abcd1234',
```



```
    resourceId: 'bbb222',
    httpMethod: 'GET',
    pathWithQueryString: '/pets/3',
  });
  try {
    const results = await apig.send(command)
    console.log(results)
  } catch (err) {
    console.log("The test on 'GET /pets/{petId}' method failed:\n", err)
  }
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        pathWithQueryString='/pets/3',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets/{petId}' failed: %s",
        error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
--resource-id bbb222 /
--http-method GET /
--path-with-query-string '/pets/3'
```

Nach einem erfolgreichen Test Ihrer API können Sie diese nun auf einer Stufe bereitstellen.

14. Stellen Sie anhand des folgenden Beispiels Ihre API in einer Stufe mit dem Namen `test` bereit. Nach der Bereitstellung Ihrer API in einer Stufe können API-Aufrufer nun Ihre API aufrufen.

JavaScript v3

```
import {APIGatewayClient, CreateDeploymentCommand } from "@aws-sdk/client-api-gateway";
(async function (){
  const apig = new APIGatewayClient({region:"us-east-1"});
  const command = new CreateDeploymentCommand({
    restApiId: 'abcd1234',
    stageName: 'test',
    stageDescription: 'test deployment'
  });
  try {
    const results = await apig.send(command)
    console.log("Deploying API succeeded\n", results)
  } catch (err) {
    console.log("Deploying API failed:\n", err)
  }
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_deployment(
        restApiId='ieps9b05sf',
        stageName='test',
        stageDescription='my test stage',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Error deploying stage %s.", error)
    raise
```

```
print('Deploying API succeeded')
print(result)
```

AWS CLI

```
aws apigateway create-deployment --rest-api-id abcd1234 \
  --region us-west-2 \
  --stage-name test \
  --stage-description 'Test stage' \
  --description 'First deployment'
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
  "id": "ab1c1d",
  "description": "First deployment",
  "createdDate": "2022-12-15T08:44:13-08:00"
}
```

D. h. dass Ihre API jetzt von Kunden aufgerufen werden kann. Sie können diese API testen, indem Sie die `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets`-URL in einen Browser eingeben und `abcd1234` durch den Bezeichner Ihrer API ersetzen.

Weitere Beispiele zum Erstellen oder Aktualisieren einer API mithilfe von AWS SDKs oder finden Sie unter [Aktionen für die AWS CLI Verwendung von API Gateway AWS SDKs](#).

Automatische Einrichtung Ihrer API

Anstatt Ihre API zu erstellen step-by-step, können Sie die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie OpenAPI oder Terraform verwenden CloudFormation, um Ihre API zu erstellen.

OpenAPI 3.0-Definition

Sie können eine OpenAPI-Definition in API Gateway importieren. Weitere Informationen finden Sie unter [the section called “OpenAPI”](#).

```
{
  "openapi" : "3.0.1",
```

```
"info" : {
  "title" : "Simple PetStore (OpenAPI)",
  "description" : "Demo API created using OpenAPI",
  "version" : "2024-05-24T20:39:34Z"
},
"servers" : [ {
  "url" : "{basePath}",
  "variables" : {
    "basePath" : {
      "default" : "Prod"
    }
  }
} ],
"paths" : {
  "/pets" : {
    "get" : {
      "responses" : {
        "200" : {
          "description" : "200 response",
          "content" : { }
        }
      }
    },
    "x-amazon-apigateway-integration" : {
      "type" : "http",
      "httpMethod" : "GET",
      "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
      "responses" : {
        "default" : {
          "statusCode" : "200"
        }
      }
    },
    "passthroughBehavior" : "when_no_match",
    "timeoutInMillis" : 29000
  }
},
"/pets/{petId}" : {
  "get" : {
    "parameters" : [ {
      "name" : "petId",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    }
  ]
}
```

```

    }
  } ],
  "responses" : {
    "200" : {
      "description" : "200 response",
      "content" : { }
    }
  },
  "x-amazon-apigateway-integration" : {
    "type" : "http",
    "httpMethod" : "GET",
    "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "responses" : {
      "default" : {
        "statusCode" : "200"
      }
    },
    "requestParameters" : {
      "integration.request.path.id" : "method.request.path.petId"
    },
    "passthroughBehavior" : "when_no_match",
    "timeoutInMillis" : 29000
  }
}
},
"components" : { }
}

```

AWS CloudFormation Vorlage

Informationen zum Bereitstellen Ihrer CloudFormation Vorlage finden Sie unter [Einen Stack auf der AWS CloudFormation Konsole](#) erstellen.

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: Simple PetStore (AWS CloudFormation)
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:

```

```
    RestApiId: !Ref Api
    ParentId: !GetAtt Api.RootResourceId
    PathPart: 'pets'
  PetIdResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !Ref PetsResource
      PathPart: '{petId}'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PetIdMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetIdResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestParameters:
        method.request.path.petId: true
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}
        RequestParameters:
          integration.request.path.id: method.request.path.petId
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  ApiDeployment:
```

```

Type: 'AWS::ApiGateway::Deployment'
DependsOn:
  - PetsMethodGet
Properties:
  RestApiId: !Ref Api
  StageName: Prod
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/Prod'

```

Terraform-Konfiguration

Weitere Informationen zu Terraform finden Sie unter [Terraform](#).

```

provider "aws" {
  region = "us-east-1" # Update with your desired region
}

resource "aws_api_gateway_rest_api" "Api" {
  name          = "Simple PetStore (Terraform)"
  description   = "Demo API created using Terraform"
}

resource "aws_api_gateway_resource" "petsResource"{
  rest_api_id = aws_api_gateway_rest_api.Api.id
  parent_id   = aws_api_gateway_rest_api.Api.root_resource_id
  path_part   = "pets"
}

resource "aws_api_gateway_resource" "petIdResource"{
  rest_api_id = aws_api_gateway_rest_api.Api.id
  parent_id   = aws_api_gateway_resource.petsResource.id
  path_part   = "{petId}"
}

resource "aws_api_gateway_method" "petsMethodGet" {
  rest_api_id    = aws_api_gateway_rest_api.Api.id
  resource_id    = aws_api_gateway_resource.petsResource.id
  http_method    = "GET"
  authorization   = "NONE"
}

resource "aws_api_gateway_method_response" "petsMethodResponseGet" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  resource_id = aws_api_gateway_resource.petsResource.id
  http_method = aws_api_gateway_method.petsMethodGet.http_method
}

```

```
        status_code = "200"
    }

    resource "aws_api_gateway_integration" "petsIntegration" {
        rest_api_id = aws_api_gateway_rest_api.Api.id
        resource_id = aws_api_gateway_resource.petsResource.id
        http_method = aws_api_gateway_method.petsMethodGet.http_method
        type        = "HTTP"

        uri          = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets"
        integration_http_method = "GET"
        depends_on    = [aws_api_gateway_method.petsMethodGet]
    }

    resource "aws_api_gateway_integration_response" "petsIntegrationResponse" {
        rest_api_id = aws_api_gateway_rest_api.Api.id
        resource_id = aws_api_gateway_resource.petsResource.id
        http_method = aws_api_gateway_method.petsMethodGet.http_method
        status_code = aws_api_gateway_method_response.petsMethodResponseGet.status_code
    }

    resource "aws_api_gateway_method" "petIdMethodGet" {
        rest_api_id    = aws_api_gateway_rest_api.Api.id
        resource_id    = aws_api_gateway_resource.petIdResource.id
        http_method    = "GET"
        authorization  = "NONE"
        request_parameters = {"method.request.path.petId" = true}
    }

    resource "aws_api_gateway_method_response" "petIdMethodResponseGet" {
        rest_api_id = aws_api_gateway_rest_api.Api.id
        resource_id = aws_api_gateway_resource.petIdResource.id
        http_method = aws_api_gateway_method.petIdMethodGet.http_method
        status_code = "200"
    }

    resource "aws_api_gateway_integration" "petIdIntegration" {
        rest_api_id = aws_api_gateway_rest_api.Api.id
        resource_id = aws_api_gateway_resource.petIdResource.id
        http_method = aws_api_gateway_method.petIdMethodGet.http_method
        type        = "HTTP"
```



```

    uri                        = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets/{id}"
    integration_http_method = "GET"
    request_parameters = {"integration.request.path.id" = "method.request.path.petId"}
    depends_on          = [aws_api_gateway_method.petIdMethodGet]
}

resource "aws_api_gateway_integration_response" "petIdIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code = aws_api_gateway_method_response.petIdMethodResponseGet.status_code
}

resource "aws_api_gateway_deployment" "Deployment" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    depends_on =
    [aws_api_gateway_integration.petsIntegration,aws_api_gateway_integration.petIdIntegration ]
}

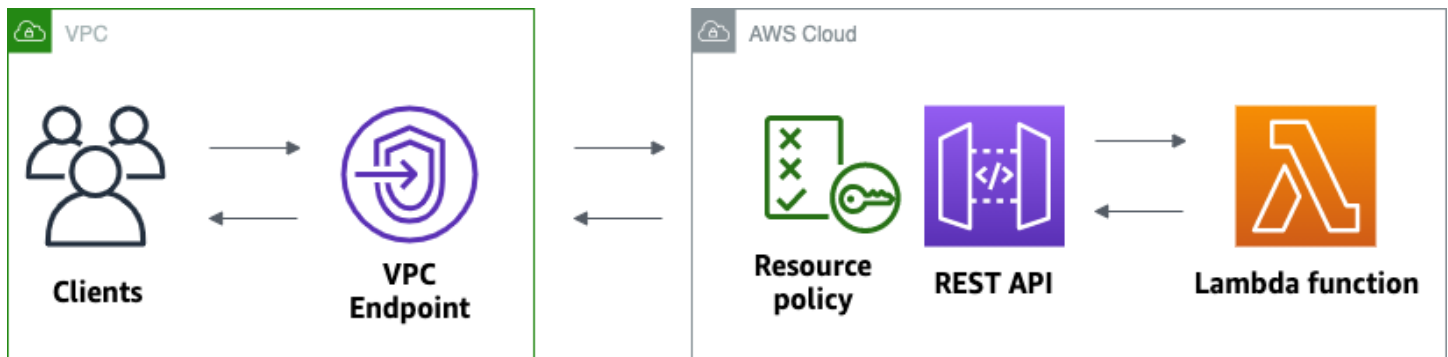
resource "aws_api_gateway_stage" "Stage" {
    stage_name      = "Prod"
    rest_api_id     = aws_api_gateway_rest_api.Api.id
    deployment_id   = aws_api_gateway_deployment.Deployment.id
}

```

Tutorial: Erstellen einer privaten REST-API

In diesem Tutorial erstellen Sie eine private REST-API. Clients können nur von Ihrer Amazon VPC aus auf die API zugreifen. Die API ist vom öffentlichen Internet isoliert, was eine übliche Sicherheitsanforderung ist.

Dieses Tutorial nimmt ungefähr 30 Minuten in Anspruch. Zunächst verwenden Sie eine CloudFormation Vorlage, um eine Amazon VPC, einen VPC-Endpunkt und eine AWS Lambda Funktion zu erstellen und eine Amazon EC2 EC2-Instance zu starten, mit der Sie Ihre API testen werden. Als Nächstes verwenden Sie die, AWS-Managementkonsole um eine private API zu erstellen und eine Ressourcenrichtlinie anzuhängen, die den Zugriff nur von Ihrem VPC-Endpunkt aus erlaubt. Zuletzt testen Sie Ihre API.



Um dieses Tutorial abzuschließen, benötigen Sie ein AWS Konto und einen AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

In diesem Tutorial wird AWS-Managementkonsole verwendet. Eine CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter [template.yaml](#).

Topics

- [Schritt 1: Erstellen von Abhängigkeiten](#)
- [Schritt 2: Erstellen einer privaten API](#)
- [Schritt 3: Erstellen Sie eine Methode und Integration](#)
- [Schritt 4: Anhängen einer Ressourcenrichtlinie](#)
- [Schritt 5: Bereitstellen Ihrer API](#)
- [Schritt 6: Stellen Sie sicher, dass Ihre API nicht öffentlich zugänglich ist](#)
- [Schritt 7: Verbinden Sie sich mit einer Instance in Ihrer VPC und rufen Sie Ihre API auf](#)
- [Schritt 8: Bereinigen](#)
- [Nächste Schritte: Automatisieren Sie mit CloudFormation](#)

Schritt 1: Erstellen von Abhängigkeiten

[Laden Sie diese Vorlage herunter und entpacken Sie sie. CloudFormation](#) Sie verwenden die Vorlage, um alle Abhängigkeiten für Ihre private API zu erstellen, einschließlich einer Amazon VPC, eines VPC-Endpunkts und einer Lambda-Funktion, die als Backend Ihrer API dient. Sie erstellen die private API später.

Um einen Stapel zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.

3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **private-api-tutorial** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt die Abhängigkeiten für Ihre API bereit, was einige Minuten dauern kann. Wenn der Status Ihres CloudFormation Stacks CREATE_COMPLETE lautet, wählen Sie Outputs. Notieren Sie sich Ihre VPC-Endpoint-ID. Sie benötigen sie für spätere Schritte in diesem Tutorial.

Schritt 2: Erstellen einer privaten API

Sie erstellen eine private API, um nur Clients in Ihrer VPC den Zugriff darauf zu ermöglichen.

Erstellen einer privaten API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie API erstellen und dann für Rest-API die Option Erstellen aus.
3. Geben Sie unter API name (API-Name) **private-api-tutorial** ein.
4. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
5. Geben Sie für VPC-Endpoint IDs die VPC-Endpoint-ID aus den Ausgaben Ihres CloudFormation Stacks ein.
6. Wählen Sie bei IP address type (IP-Adresstyp) Dualstack.
7. Wählen Sie Create API (API erstellen) aus.

Schritt 3: Erstellen Sie eine Methode und Integration

Sie erstellen eine GET Methode und eine Lambda-Integration, um GET-Anfragen an Ihre API zu bearbeiten. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion, die Sie in Schritt 1 erstellt haben und gibt dann eine Antwort an den Client zurück.

So erstellen Sie eine Methode und Integration

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Methode erstellen aus.
4. Für Methodentyp wählen Sie GET aus.
5. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
6. Schalten Sie die Lambda-Proxy-Integration ein. Mit einer Lambda-Proxy-Integration sendet API Gateway ein Ereignis mit einer definierten Struktur an Lambda und transformiert die Antwort von Ihrer Lambda-Funktion in eine HTTP-Antwort.
7. Wählen Sie für die Lambda-Funktion die Funktion aus, die Sie mit der CloudFormation Vorlage in Schritt 1 erstellt haben. Der Name der Funktion beginnt mit **private-api-tutorial**.
8. Wählen Sie Methode erstellen aus.

Schritt 4: Anhängen einer Ressourcenrichtlinie

Sie hängen Ihrer API eine [Ressourcenrichtlinie](#) an, die es Clients ermöglicht, Ihre API nur über Ihren VPC-Endpunkt aufzurufen. Um den Zugriff auf Ihre API weiter einzuschränken, können Sie auch eine [VPC-Endpunktrichtlinie](#) für Ihren VPC-Endpunkt konfigurieren, aber das ist für dieses Tutorial nicht erforderlich.

So hängen Sie eine Ressourcenrichtlinie an

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Ressourcenrichtlinie aus und klicken Sie dann auf Richtlinie erstellen.
4. Geben Sie die folgende Richtlinie ein: *vpceID* Ersetzen Sie es durch Ihre VPC-Endpunkt-ID aus den Ausgaben Ihres CloudFormation Stacks.

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "Principal": "*",
    "Action": "execute-api:Invoke",
    "Resource": "execute-api:/*",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-abcd1234"
      }
    }
  },
  {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "execute-api:Invoke",
    "Resource": "execute-api:/*"
  }
]
```

5. Wählen Sie **Änderungen speichern** aus.

Schritt 5: Bereitstellen Ihrer API

Als Nächstes stellen Sie Ihre API bereit, um sie Clients in Ihrer Amazon VPC zur Verfügung zu stellen.

So stellen Sie eine API bereit

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Klicken Sie auf **Deploy API**.
4. Wählen Sie für Stufe die Option **Neue Stufe** aus.
5. Geben Sie für Stage name (Stufenname) **test** ein.
6. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
7. Wählen Sie **Deploy (Bereitstellen)** aus.

Jetzt sind Sie bereit, Ihre API zu testen.

Schritt 6: Stellen Sie sicher, dass Ihre API nicht öffentlich zugänglich ist

Verwenden Sie, `curl` um zu überprüfen, ob Sie Ihre API nicht außerhalb Ihrer Amazon VPC aufrufen können.

So testen Sie Ihre API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie im Hauptnavigationsbereich Stufen aus und klicken Sie dann auf die Stufe Test.
4. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren. Die URL sieht folgendermaßen aus `https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. Für den VPC-Endpunkt, den Sie in Schritt 1 erstellt haben, ist `private` DNS aktiviert, sodass Sie die bereitgestellte URL zum Aufrufen Ihrer API verwenden können.
5. Verwenden Sie `curl`, um zu versuchen, Ihre API von außerhalb Ihrer VPC aufzurufen.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Curl zeigt an, dass der Endpunkt Ihrer API nicht aufgelöst werden kann. Wenn Sie eine andere Antwort erhalten, gehen Sie zurück zu Schritt 2 und stellen Sie sicher, dass Sie `Private` für den Endpunkttyp Ihrer API auswählen.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Als Nächstes stellen Sie eine Verbindung zu einer Amazon-EC2-Instance in Ihrer VPC her, um Ihre API aufzurufen.

Schritt 7: Verbinden Sie sich mit einer Instance in Ihrer VPC und rufen Sie Ihre API auf

Als Nächstes testen Sie Ihre API aus Ihrer Amazon VPC heraus. Um auf Ihre private API zuzugreifen, stellen Sie eine Verbindung zu einer Amazon-EC2-Instance in Ihrer VPC her und verwenden

dann curl, um Ihre API aufzurufen. Sie verwenden Systems-Manager Session-Manager, um eine Verbindung mit Ihrer Instance im Browser herzustellen.

So testen Sie Ihre API

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie Instances.
3. Wählen Sie den Instanznamen aus private-api-tutorial, den Sie mit der CloudFormation Vorlage in Schritt 1 erstellt haben.
4. Wählen Sie Verbinden und wählen Sie dann Session-Manager.
5. Wählen Sie Verbinden , um eine browserbasierte Sitzung für Ihre Instance zu starten.
6. Verwenden Sie in Ihrer Session-Manager-Sitzung curl, um Ihre API aufzurufen. Sie können Ihre API aufrufen, da Sie eine Instance in Ihrer Amazon VPC verwenden.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Stellen Sie sicher, dass Sie die Antwort erhalte Hello from Lambda!.



Sie haben erfolgreich eine API erstellt, auf die nur von Ihrer Amazon VPC aus zugegriffen werden kann und dann überprüft, dass sie funktioniert.

Schritt 8: Bereinigen

Um unnötige Kosten zu vermeiden, löschen Sie die Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben. Mit den folgenden Schritten werden Ihre REST-API und Ihr CloudFormation Stack gelöscht.

So löschen Sie eine REST-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIsSeite eine API aus. Wählen Sie API-Aktionen aus, klicken Sie auf API löschen und bestätigen Sie anschließend Ihre Auswahl.

Um einen CloudFormation Stapel zu löschen

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Nächste Schritte: Automatisieren Sie mit CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine vollständige CloudFormation -Beispielvorlage finden Sie unter [template.yaml](#).

Amazon API Gateway-HTTP-API-Tutorials

Die folgenden Tutorials bieten praktische Übungen, die Ihnen beim Kennenlernen von API Gateway HTTP APIs helfen.

Themen

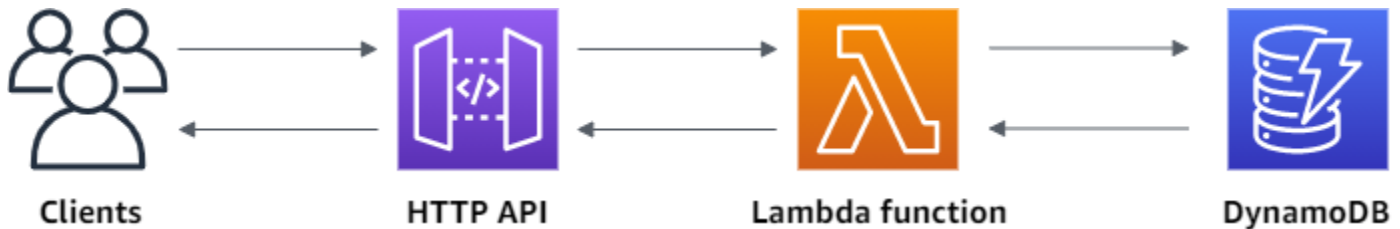
- [Tutorial: Erstellen einer CRUD-HTTP-API mit Lambda und DynamoDB](#)
- [Tutorial: Erstellen einer HTTP-API mit privater Integration für einen Amazon-ECS-Service](#)

Tutorial: Erstellen einer CRUD-HTTP-API mit Lambda und DynamoDB

In diesem Tutorial erstellen Sie eine serverlose API, die Elemente aus einer DynamoDB-Tabelle erstellt, liest, aktualisiert und löscht. DynamoDB ist ein vollständig verwalteter NoSQL-Datenbankservice, der schnelle und vorhersehbare Leistung nahtlos skalierbar bereitstellt. Dieses Tutorial dauert ungefähr 30 Minuten, und Sie können es im Rahmen des [kostenlosen Kontingents von AWS](#) durchgehen.

Zunächst erstellen Sie mit der DynamoDB-Konsole eine [DynamoDB-Tabelle](#). Dann erstellen Sie mit der AWS Lambda Konsole eine [Lambda-Funktion](#). Als Nächstes erstellen Sie eine HTTP-API mit der API-Gateway-Konsole. Zuletzt testen Sie Ihre API.

Wenn Sie Ihre HTTP-API aufrufen, leitet API Gateway die Anfrage an Ihre Lambda-Funktion weiter. Die Lambda-Funktion interagiert mit DynamoDB und gibt eine Antwort an API Gateway zurück. API Gateway gibt dann eine Antwort an Sie zurück.



Um diese Übung abzuschließen, benötigen Sie ein AWS Konto und einen AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

In diesem Tutorial wird AWS-Managementkonsole verwendet. Eine AWS SAM Vorlage, mit der diese API und alle zugehörigen Ressourcen erstellt werden, finden Sie unter [template.yaml](#).

Topics

- [Schritt 1: Erstellen einer DynamoDB-Tabelle](#)
- [Schritt 2: Erstellen einer Lambda-Funktion](#)
- [Schritt 3: HTTP-API erstellen](#)
- [Schritt 4: Routen erstellen](#)
- [Schritt 5: Eine Integration erstellen](#)
- [Schritt 6: Ihre Integration an Routen anfügen](#)
- [Schritt 7: Ihre API testen](#)
- [Schritt 8: Bereinigen](#)
- [Nächste Schritte: Automatisieren Sie mit AWS SAM oder CloudFormation](#)

Schritt 1: Erstellen einer DynamoDB-Tabelle

Sie verwenden eine [DynamoDB-Tabelle](#), um Daten für Ihre API zu speichern.

Jedes Element hat eine eindeutige ID, die wir als [Partitionsschlüssel](#) für die Tabelle verwenden.

So erstellen Sie eine DynamoDB-Tabelle

1. Öffnen Sie die DynamoDB-Konsole unter <https://console.aws.amazon.com/dynamodb/>
2. Wählen Sie Create table aus.
3. Geben Sie für Table name (Tabellenname) **http-crud-tutorial-items** ein.
4. Geben Sie unter Partitionsschlüssel den Wert **id** ein.
5. Wählen Sie Create table (Tabelle erstellen) aus.

Schritt 2: Erstellen einer Lambda-Funktion

Sie erstellen eine [Lambda](#)-Funktion für das Backend Ihrer API. Diese Lambda-Funktion erstellt, liest, aktualisiert und löscht Elemente aus DynamoDB. Die Funktion verwendet [Ereignisse von API Gateway](#), um zu bestimmen, wie mit DynamoDB interagiert wird. Der Einfachheit halber verwendet dieses Tutorial eine einzige Lambda-Funktion. Als Best Practice sollten Sie separate Funktionen für jede Route erstellen. Weitere Informationen finden Sie unter [Der Lambda-Monolith](#).

So erstellen Sie eine Lambda-Funktion

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie Create function (Funktion erstellen).
3. Geben Sie für Function name (Funktionsname) **http-crud-tutorial-function** ein.
4. Wählen Sie für Laufzeit die neueste unterstützte Node.js- oder Python-Laufzeit aus.
5. Wählen Sie unter Berechtigungen die Option Standard-Ausführungsrolle ändern aus.
6. Wählen Sie Neue Rolle aus Richtlinienvorlagen erstellen aus AWS .
7. Geben Sie für Role name (Rollenname) den Namen **http-crud-tutorial-role** ein.
8. Wählen Sie für Policy templates (Richtlinienvorlagen) **Simple microservice permissions** aus. Diese Richtlinie gewährt der Lambda-Funktion die Berechtigung, mit DynamoDB zu interagieren.

Note

Dieses Tutorial verwendet der Einfachheit halber eine verwaltete Richtlinie. Als Best Practice sollten Sie Ihre eigene IAM-Richtlinie erstellen, um die erforderlichen Mindestberechtigungen zu gewähren.

9. Wählen Sie Create function (Funktion erstellen).

10. Öffnen Sie die Lambda-Funktion im Code-Editor der Konsole und ersetzen Sie den Inhalt durch den folgenden Code. Wählen Sie Deploy (Bereitstellen) aus, um Ihre Funktion zu aktualisieren.

Node.js

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "http-crud-tutorial-items";

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
      case "DELETE /items/{id}":
        await dynamo.send(
          new DeleteCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = `Deleted item ${event.pathParameters.id}`;
        break;
      case "GET /items/{id}":
        body = await dynamo.send(
          new GetCommand({
```

```
        TableName: tableName,
        Key: {
            id: event.pathParameters.id,
        },
    })
    );
    body = body.Item;
    break;
case "GET /items":
    body = await dynamo.send(
        new ScanCommand({ TableName: tableName })
    );
    body = body.Items;
    break;
case "PUT /items":
    let requestJSON = JSON.parse(event.body);
    await dynamo.send(
        new PutCommand({
            TableName: tableName,
            Item: {
                id: requestJSON.id,
                price: requestJSON.price,
                name: requestJSON.name,
            },
        })
    );
    body = `Put item ${requestJSON.id}`;
    break;
default:
    throw new Error(`Unsupported route: "${event.routeKey}"`);
}
} catch (err) {
    statusCode = 400;
    body = err.message;
} finally {
    body = JSON.stringify(body);
}

return {
    statusCode,
    body,
    headers,
};
```

```
};
```

Python

```
import json
import boto3
from decimal import Decimal

client = boto3.client('dynamodb')
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table('http-crud-tutorial-items')
tableName = 'http-crud-tutorial-items'

def lambda_handler(event, context):
    print(event)
    body = {}
    statusCode = 200
    headers = {
        "Content-Type": "application/json"
    }

    try:
        if event['routeKey'] == "DELETE /items/{id}":
            table.delete_item(
                Key={'id': event['pathParameters']['id']})
            body = 'Deleted item ' + event['pathParameters']['id']
        elif event['routeKey'] == "GET /items/{id}":
            body = table.get_item(
                Key={'id': event['pathParameters']['id']})
            body = body["Item"]
            responseBody = [
                {'price': float(body['price']), 'id': body['id'], 'name':
body['name']}]
            body = responseBody
        elif event['routeKey'] == "GET /items":
            body = table.scan()
            body = body["Items"]
            print("ITEMS----")
            print(body)
            responseBody = []
            for items in body:
                responseItems = [
```

```

        {'price': float(items['price']), 'id': items['id'], 'name':
items['name']}]
        responseBody.append(responseItems)
        body = responseBody
    elif event['routeKey'] == "PUT /items":
        requestJSON = json.loads(event['body'])
        table.put_item(
            Item={
                'id': requestJSON['id'],
                'price': Decimal(str(requestJSON['price'])),
                'name': requestJSON['name']
            })
        body = 'Put item ' + requestJSON['id']
    except KeyError:
        statusCode = 400
        body = 'Unsupported route: ' + event['routeKey']
    body = json.dumps(body)
    res = {
        "statusCode": statusCode,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": body
    }
    return res

```

Schritt 3: HTTP-API erstellen

Die HTTP-API bietet einen HTTP-Endpunkt für Ihre Lambda-Funktion. In diesem Schritt erstellen Sie eine leere API. In den folgenden Schritten konfigurieren Sie Routen und Integrationen, um Ihre API und Ihre Lambda-Funktion zu verbinden.

So erstellen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) und dann für HTTP API die Option Build (Erstellen) aus.
3. Geben Sie unter API name (API-Name) **http-crud-tutorial-api** ein.
4. Wählen Sie als IP-Adresstyp die Option. IPv4

5. Wählen Sie Next.
6. Wählen Sie für Configure routes (Routen konfigurieren) die Option Next (Weiter) aus, um die Routenerstellung zu überspringen. Sie werden Routen später erstellen.
7. Überprüfen Sie die Phase, die API Gateway für Sie erstellt, und wählen Sie dann Next (Weiter) aus.
8. Wählen Sie Create aus.

Schritt 4: Routen erstellen

Routen sind eine Möglichkeit, eingehende API-Anforderungen an Backend-Ressourcen zu senden. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispiel, GET /items. Für diese Beispiel-API erstellen wir vier Routen:

- GET /items/{id}
- GET /items
- PUT /items
- DELETE /items/{id}

So werden Routen erstellt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Routes (Routen) aus.
4. Wählen Sie Create aus.
5. Wählen Sie für Methode **GET** aus.
6. Für den Pfad geben Sie **/items/{id}**. {id} am Ende des Pfads ist ein Pfadparameter, den API Gateway vom Anforderungspfad abrufen, wenn ein Client eine Anforderung stellt.
7. Wählen Sie Create aus.
8. Wiederholen Sie die Schritte 4-7 für GET /items, DELETE /items/{id} und PUT /items.

API Gateway > APIs > Routes - http-crud-tutorial-api (acbd1234)

Routes

Stage: - Deploy

Routes for http-crud-tutorial-api

Create

Search

- ▼ /items
 - GET
 - PUT**
- ▼ /{id}
 - GET
 - DELETE

Route details

PUT /items (ID: bdlb4q)

Delete Edit

ARN
arn:aws:apigateway:us-west-1::apis/abcd1234/routes/aaa111

Authorization
Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open.

No authorizer attached to this route. Attach authorization

Integration
The integration is the backend resource that this route calls when it receives a request.

No integration attached to this route. Attach integration

Schritt 5: Eine Integration erstellen

Sie erstellen eine Integration, um eine Route mit Backend-Ressourcen zu verbinden. Für diese Beispiel-API erstellen Sie eine Lambda-Integration, die Sie für alle Routen verwenden.

So wird eine Integration erstellt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wähle Sie Integrations (Integrationen) aus.
4. Wählen Sie Manage integrations (Integrationen verwalten) und dann Create (Erstellen) aus.
5. Überspringen Sie Attach this integration to a route (Diese Integration an eine Route anfügen). Dies werden Sie in einem späteren Schritt abschließen.
6. Für Integration type (Integrationstyp) wählen Sie Lambda Function (Lambda-Funktion) aus.
7. Geben Sie für die Lambda function (Lambda-Funktion) **http-crud-tutorial-function** ein.
8. Wählen Sie Create aus.

Schritt 6: Ihre Integration an Routen anfügen

Für diese Beispiel-API verwenden Sie dieselbe Lambda-Integration für alle Routen. Nachdem Sie die Integration allen Routen der API angefügt haben, wird Ihre Lambda-Funktion aufgerufen, wenn ein Client eine Ihrer Routen aufruft.

So werden Integrationen an Routen angefügt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wähle Sie Integrations (Integrationen) aus.
4. Wählen Sie eine Route aus.
5. Wählen Sie unter Choose an existing integration (Eine vorhandene Integration auswählen) **http-crud-tutorial-function** aus.
6. Wählen Sie Attach integration (Integration anhängen) aus.
7. Wiederholen Sie die Schritte 4 bis 6 für alle Routen.

Alle Routen zeigen, dass eine AWS Lambda Integration angehängt ist.

API Gateway > APIs > [http-crud-tutorial-api \(abdc1234\)](#) > Integrations

Integrations

Stage: - ▼

Deploy

Attach integrations to routes

Manage integrations

Routes for http-crud-tutorial-api

Search

▼ /items

GET AWS Lambda

PUT AWS Lambda

▼ /{id}

GET AWS Lambda

DELETE AWS Lambda

Integration details for route

Detach integration

Manage integration

PUT /items (aaa111)

Lambda function

http-crud-tutorial-function (us-west-1) [↗](#)

Integration ID

bbb222

Description

-

Payload format version

The parsing algorithm for the payload sent to and returned from your Lambda function. [Learn more.](#)

2.0 (interpreted response format)

Invoke permissions

The resource policy of the Lambda function determines if API Gateway can invoke it. You can run the AWS CLI command snippet below to give API Gateway permission to invoke your AWS Lambda function.

► Example policy statement

Timeout

The number of milliseconds that API Gateway should wait for a response from the integration before timing out.

30000

Request parameter mapping

Not configured

Response parameter mappings

Not configured

Jetzt, da Sie eine HTTP-API mit Routen und Integrationen haben, können Sie Ihre API testen.

Schritt 7: Ihre API testen

Um sicherzustellen, dass Ihre API funktioniert, verwenden Sie [curl](#).

So wird die URL zum Aufrufen Ihrer API abgerufen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Notieren Sie sich die Aufruf-URL Ihrer API. Sie wird auf der Seite Details unter Invoke URL (URL aufrufen) angezeigt.

API Gateway > APIs > http-crud-tutorial-api (abcd1234)

http-crud-tutorial-api

Stage: - ▼

Deploy

API details

Edit

API ID abcd1234	Protocol HTTP	Created 2024-05-02
Description My HTTP CRUD tutorial API	Default endpoint ✔ Enabled https://abcd1234.execute-api.us-west-1.amazonaws.com	ARN arn:aws:apigateway:us-west-1::apis/abcd1234

Stages for http-crud-tutorial-api (1)

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://abcd1234.execute-api.us-west-1.amazonaws.com	abc000	enabled	2024-05-02

4. Kopieren Sie die Aufruf-URL Ihrer API.

Die vollständige URL sieht folgendermaßen aus: `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Erstellen oder Aktualisieren eines Elements

- Verwenden Sie den folgenden Befehl, um ein Element zu erstellen oder zu aktualisieren. Der Befehl enthält einen Anfragetext mit der ID, dem Preis und dem Namen des Elements.

```
curl -X "PUT" -H "Content-Type: application/json" -d '{"id": "123",
  "price": 12345, "name": "myitem"}' https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Abrufen aller Elemente

- Verwenden Sie den folgenden Befehl, um alle Elemente aufzulisten.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Abrufen eines Elements

- Verwenden Sie den folgenden Befehl, um ein Element anhand seiner ID zu erhalten.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

So löschen Sie einen Artikel

1. Verwenden Sie den folgenden Befehl, um ein Element zu löschen.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Rufen Sie alle Elemente ab, um zu überprüfen, ob das Element gelöscht wurde.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Schritt 8: Bereinigen

Um unnötige Kosten zu vermeiden, löschen Sie die Ressourcen, die Sie im Rahmen dieser Übung „Erste Schritte“ erstellt haben. In den folgenden Schritten werden Ihre HTTP-API, Ihre Lambda-Funktion und die zugehörigen Ressourcen gelöscht.

So löschen Sie eine DynamoDB-Tabelle

1. Öffnen Sie die DynamoDB-Konsole unter <https://console.aws.amazon.com/dynamodb/>
2. Wählen Sie Ihre Tabelle aus.
3. Wählen Sie Delete Table (Tabelle löschen).
4. Bestätigen Sie Ihre Wahl und wählen Sie Delete (Löschen) aus.

So löschen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIsSeite eine API aus. Wählen Sie Actions und anschließend Delete.
3. Wählen Sie Delete (Löschen).

So löschen Sie eine Lambda-Funktion

1. Melden Sie sich bei der Lambda-Konsole unter <https://console.aws.amazon.com/lambda> an.
2. Wählen Sie auf der Seite Functions (Funktionen) eine Funktion aus. Wählen Sie Actions und anschließend Delete.
3. Wählen Sie Delete (Löschen).

So löschen Sie die Protokollgruppe einer Lambda-Funktion

1. Öffnen Sie in der CloudWatch Amazon-Konsole die [Seite Protokollgruppen](#).
2. Wählen Sie auf der Seite Log groups (Protokollgruppen) die Protokollgruppe (/aws/lambda/http-crud-tutorial-function) der Funktion aus. Wählen Sie Actions (Aktionen) und dann Delete log group (Protokollgruppe löschen) aus.
3. Wählen Sie Delete (Löschen).

So löschen Sie die Ausführungsrolle einer Lambda-Funktion

1. Öffnen Sie in der AWS Identity and Access Management Konsole die [Seite Rollen](#).
2. Wählen Sie die Rolle der Funktion aus, zum Beispiel, http-crud-tutorial-role.
3. Wählen Sie Delete role (Rolle löschen) aus.
4. Wählen Sie Yes, delete (Ja, löschen) aus.

Nächste Schritte: Automatisieren Sie mit AWS SAM oder CloudFormation

Sie können die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie CloudFormation oder AWS SAM verwenden. Ein Beispiel für eine AWS SAM -Vorlage für dieses Tutorial finden Sie unter [template.yaml](#).

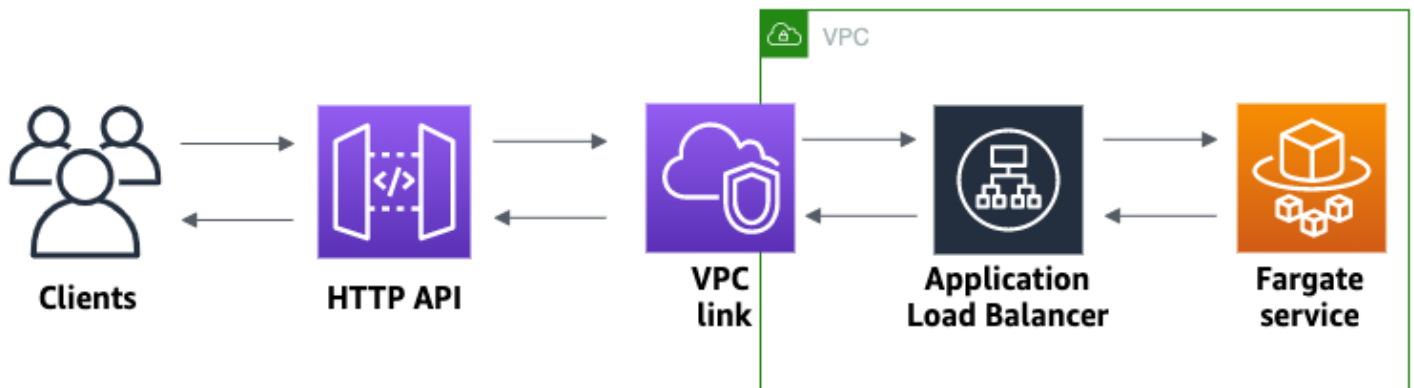
CloudFormation Beispielvorlagen finden Sie unter [CloudFormation Beispielvorlagen](#).

Tutorial: Erstellen einer HTTP-API mit privater Integration für einen Amazon-ECS-Service

In diesem Tutorial erstellen Sie eine serverlose API, die eine Verbindung zu einem Amazon-ECS-Service herstellt, der in einer Amazon VPC ausgeführt wird. Clients außerhalb Ihrer Amazon VPC können die API verwenden, um auf Ihren Amazon-ECS-Service zuzugreifen.

Für dieses Tutorial brauchen Sie ungefähr eine Stunde. Zunächst verwenden Sie eine CloudFormation Vorlage, um einen Amazon VPC- und Amazon ECS-Service zu erstellen. Dann verwenden Sie die API-Gateway-Konsole, um einen VPC-Link zu erstellen. Der VPC-Link ermöglicht es API Gateway, auf den Amazon-ECS-Service zuzugreifen, der in Ihrer Amazon VPC ausgeführt wird. Als Nächstes erstellen Sie eine HTTP-API, die den VPC-Link verwendet, um eine Verbindung zu Ihrem Amazon-ECS-Service herzustellen. Zuletzt testen Sie Ihre API.

Wenn Sie Ihre HTTP-API aufrufen, leitet API Gateway die Anfrage über Ihren VPC-Link an Ihren Amazon-ECS-Service weiter und gibt dann die Antwort vom Service zurück.



Um dieses Tutorial abzuschließen, benötigen Sie ein AWS Konto und einen AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

In diesem Tutorial wird AWS-Managementkonsole verwendet. Eine CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter [template.yaml](#).

Topics

- [Schritt 1: Erstellen Sie einen Amazon-ECS-Service](#)
- [Schritt 2: Erstellen eines VPC-Links](#)
- [Schritt 3: HTTP-API erstellen](#)
- [Schritt 4: Erstellen einer Route](#)
- [Schritt 5: Eine Integration erstellen](#)
- [Schritt 6: Ihre API testen](#)
- [Schritt 7: Bereinigen](#)
- [Nächste Schritte: Automatisieren Sie mit CloudFormation](#)

Schritt 1: Erstellen Sie einen Amazon-ECS-Service

Amazon ECS ist ein Container-Management-Service, mit dem Sie unkompliziert Docker-Container in einem Cluster ausführen, beenden und verwalten können. In diesem Tutorial führen Sie Ihren Cluster auf einer serverlosen Infrastruktur aus, die von Amazon ECS verwaltet wird.

Laden Sie [diese CloudFormation Vorlage](#) herunter und entpacken Sie sie, wodurch alle Abhängigkeiten für den Service, einschließlich einer Amazon VPC, erstellt werden. Sie verwenden die Vorlage, um einen Amazon-ECS-Service zu erstellen, der einen Application Load Balancer verwendet.

Um einen Stack zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **http-api-private-integrations-tutorial** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt den ECS-Service bereit, was einige Minuten dauern kann. Wenn der Status Ihres CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen eines VPC-Links

Ein VPC-Link ermöglicht es API Gateway, auf private Ressourcen in einer Amazon VPC zuzugreifen. Sie verwenden einen VPC-Link, um Clients den Zugriff auf Ihren Amazon-ECS-Service über Ihre HTTP-API zu ermöglichen.

So erstellen Sie einen VPC-Link

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf VPC-Links und dann auf Erstellen.

Sie müssen möglicherweise auf das Menüsymbol klicken, um den Hauptnavigationsbereich zu öffnen.
3. Wählen Sie unter VPC-Linkversion die Option VPC-Link für HTTP aus. APIs
4. Geben Sie als Name die Zeichenfolge „**private-integrations-tutorial**“ ein.
5. Wählen Sie für VPC die in Schritt 1 erstellte VPC aus. Der Name muss mit `PrivateIntegrationsStack` beginnen.
6. Wählen Sie für Subnetze die beiden privaten Subnetze in Ihrer VPC aus. Ihre Namen enden mit `PrivateSubnet`.
7. Wählen Sie unter Sicherheitsgruppen die Gruppen-ID aus, die mit `private-integrations-tutorial` beginnt und die `PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup`-Beschreibung enthält.
8. Wählen Sie Erstellen aus.

Nachdem Sie Ihren VPC-Link erstellt haben, bietet API-Gateway-Elastic-Network-Interfaces für den Zugriff auf Ihre VPC an. Der Vorgang kann einige Minuten dauern. In der Zwischenzeit können Sie Ihre API erstellen.

Schritt 3: HTTP-API erstellen

Die HTTP-API bietet einen HTTP-Endpunkt für Ihre Amazon-ECS-Service-Funktion. In diesem Schritt erstellen Sie eine leere API. In den Schritten 4 und 5 konfigurieren Sie eine Route und eine Integration, um Ihre API und Ihren Amazon-ECS-Service miteinander zu verbinden.

So erstellen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) und dann für HTTP API die Option Build (Erstellen) aus.
3. Geben Sie unter API name (API-Name) **http-private-integrations-tutorial** ein.

4. Wählen Sie als IP-Adresstyp. IPv4
5. Wählen Sie Next.
6. Wählen Sie für Configure routes (Routen konfigurieren) die Option Next (Weiter) aus, um die Routenerstellung zu überspringen. Sie werden Routen später erstellen.
7. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. API Gateway erstellt eine `$default` Phase mit aktivierten automatischen Bereitstellungen, was die beste Wahl für dieses Tutorial ist. Wählen Sie Next.
8. Wählen Sie Create aus.

Schritt 4: Erstellen einer Route

Routen sind eine Möglichkeit, eingehende API-Anforderungen an Backend-Ressourcen zu senden. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispiel, GET `/items`. Für diese Beispiel-API erstellen wir eine Route:

Erstellen Sie eine Route.

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Routes (Routen) aus.
4. Wählen Sie Create aus.
5. Wählen Sie für Methode **ANY** aus.
6. Für den Pfad geben Sie `/{{proxy+}}`. Das `{{proxy+}}` am Ende des Pfades ist eine „gierige“ Pfadvariable. API Gateway sendet alle Anfragen an Ihre API an diese Route.
7. Wählen Sie Create aus.

Schritt 5: Eine Integration erstellen

Sie erstellen eine Integration, um eine Route mit Backend-Ressourcen zu verbinden.

So wird eine Integration erstellt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.

3. Wählen Sie Integrations (Integrationen) aus.
4. Wählen Sie Manage integrations (Integrationen verwalten) und dann Create (Erstellen) aus.
5. Wählen Sie für Diese Integration an eine Route anhängen die ANY/{proxy+}-Route aus, die Sie zuvor erstellt haben.
6. Für Integrationstyp wählen Sie Private Ressource aus.
7. Für Integrationsdetails wählen Sie Manuell auswählen.
8. Für Ziel-Service wählen Sie ALB/NLB.
9. Für Lastenverteilung wählen Sie die Lastenverteilung aus, die Sie mit der CloudFormation - Vorlage in Schritt 1 erstellt haben. Der Name sollte mit HTTP-Priva beginnen.
10. Für den Listener wählen Sie **HTTP 80**.
11. Für VPC-Link wählen Sie den VPC-Link aus, den Sie in Schritt 2 erstellt haben. Ihr Name sollte laute `private-integrations-tutorial`.
12. Wählen Sie Create aus.

Um zu überprüfen, ob Ihre Route und Integration korrekt eingerichtet ist, wählen Sie Integrationen an Routen anhängen aus. Die Konsole zeigt, dass Sie eine ANY /{proxy+} Route mit einer Integration in einen VPC-Load-Balancer haben.

Integrations

Stage: - ▼

Deploy

[Attach integrations to routes](#)[Manage integrations](#)

Routes for private-integrations-tutorial

▼ /{proxy+}

ANY

VPC Load Balancer

Integration details for route

[Detach integration](#)[Manage integration](#)

ANY /{proxy+} (abc123)

Load balancer listener

ANY HTTP:80 - http-p-Priva-ABCD1234abc



Integration ID

aaa111

Description

-

VPC link

[efg456](#)

Timeout

The number of milliseconds that API Gateway should wait for a response from the integration before timing out.

30000

Jetzt sind Sie bereit, Ihre API zu testen.

Schritt 6: Ihre API testen

Als Nächstes testen Sie Ihre API, um sicherzustellen, dass sie funktioniert. Verwenden Sie zur Vereinfachung einen Webbrowser, um Ihre API aufzurufen.

So testen Sie Ihre API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Notieren Sie sich die Aufruf-URL Ihrer API.

my-http-api

Stage: - ▼

Deploy

API details

Edit

API ID
abcd1234**Protocol**
HTTP**Created**
2023-10-30**Description**
My first HTTP API**Default endpoint**
✔ Enabled
<https://abcd1234.execute-api.us-west-1.amazonaws.com>**ARN**
[arn:aws:apigateway:us-west-1::apis/abcd1234](#)

Stages for my-http-api (1)

Find resources

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://abcd1234.execute-api.us-west-1.amazonaws.com	aaa111	enabled	2023-10-30

4. Rufen Sie in einem Webbrowser die Aufruf-URL Ihrer API auf.

Die URL sollte wie folgt aussehen: `https://abcdef123.execute-api.us-east-2.amazonaws.com`.

Ihr Browser sendet eine GET-Anforderung an die API.

5. Stellen Sie sicher, dass die Antwort Ihrer API eine Willkommensnachricht ist, die Ihnen mitteilt, dass Ihre App auf Amazon ECS ausgeführt wird.

Wenn Sie die Willkommensnachricht sehen, haben Sie erfolgreich einen Amazon-ECS-Service erstellt, der in einer Amazon VPC ausgeführt wird und Sie haben eine API-Gateway-HTTP-API mit einem VPC-Link für den Zugriff auf den Amazon-ECS-Service verwendet.

Schritt 7: Bereinigen

Um unnötige Kosten zu verhindern, löschen Sie die Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben. Mit den folgenden Schritten werden Ihr VPC-Link, Ihr CloudFormation Stack und Ihre HTTP-API gelöscht.

So löschen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIsSeite eine API aus. Wählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

So löschen Sie einen VPC-Link

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie VPC-Link.
3. Wählen Sie Ihren VPC-Link aus, wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Um einen Stapel zu löschen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Nächste Schritte: Automatisieren Sie mit CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine vollständige CloudFormation -Beispielvorlage finden Sie unter [template.yaml](#).

Amazon API Gateway-WebSocket-API-Tutorials

Die folgenden Tutorials enthalten praktische Übungen, in denen Sie mehr über WebSocket-APIs in API Gateway erfahren.

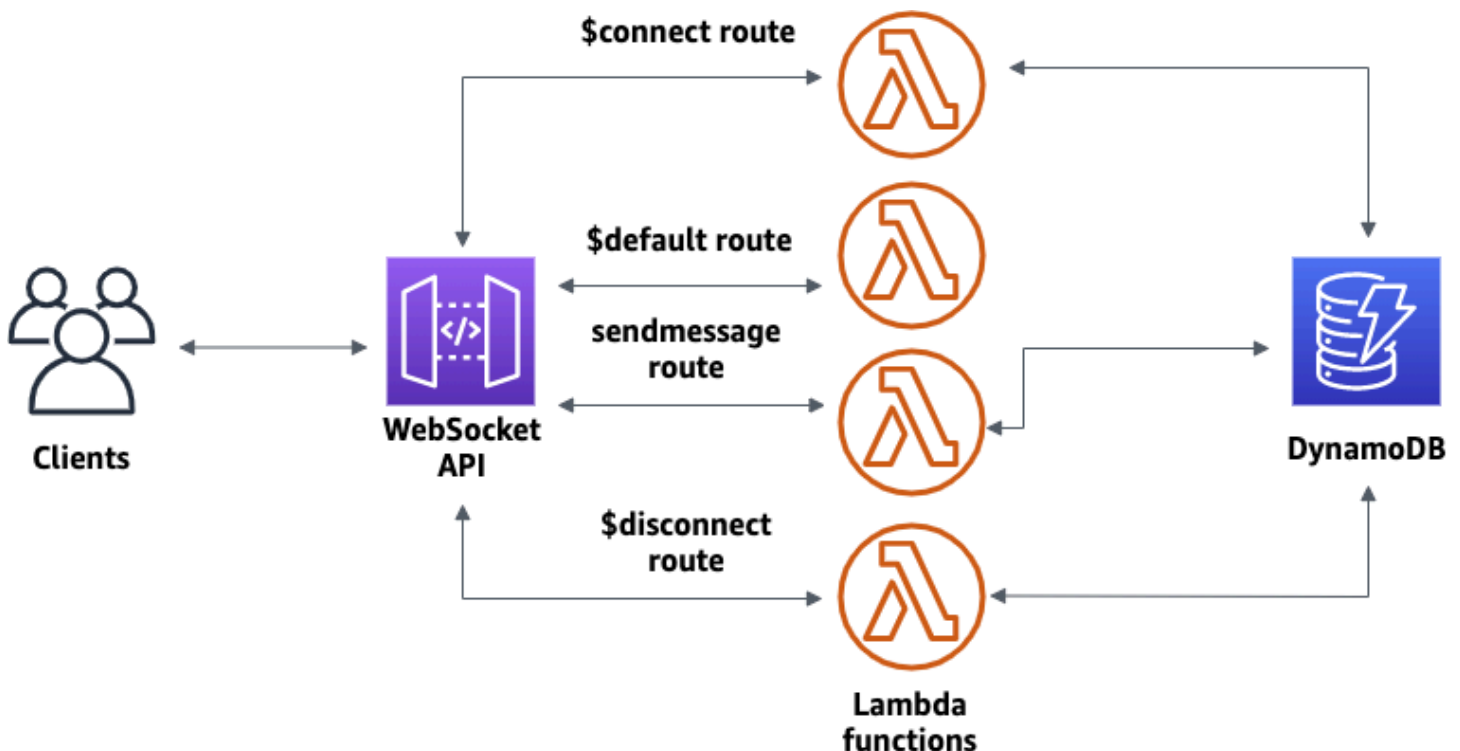
Themen

- [Tutorial: Erstellen Sie eine WebSocket Chat-App mit einer WebSocket API, Lambda und DynamoDB](#)
- [Tutorial: Erstellen Sie eine WebSocket API mit einer AWS Integration](#)

Tutorial: Erstellen Sie eine WebSocket Chat-App mit einer WebSocket API, Lambda und DynamoDB

In diesem Tutorial erstellen Sie eine serverlose Chat-Anwendung mit einer WebSocket API. Mit einer WebSocket API können Sie die bidirektionale Kommunikation zwischen Clients unterstützen. Kunden können Nachrichten erhalten, ohne Updates abfragen zu müssen.

Dieses Tutorial nimmt ungefähr 30 Minuten in Anspruch. Zunächst verwenden Sie eine CloudFormation Vorlage, um Lambda-Funktionen zu erstellen, die API-Anfragen verarbeiten, sowie eine DynamoDB-Tabelle, in der Ihr Client gespeichert wird. IDs Anschließend verwenden Sie die API Gateway Gateway-Konsole, um eine WebSocket API zu erstellen, die in Ihre Lambda-Funktionen integriert ist. Zuletzt testen Sie Ihre API, um zu prüfen, ob Nachrichten gesendet und empfangen werden.



Um dieses Tutorial abzuschließen, benötigen Sie ein AWS Konto und einen AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).

Sie benötigen außerdem `wscat`, um eine Verbindung zu Ihrer API herzustellen. Weitere Informationen finden Sie unter [the section called “Wird verwendet wscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).

Topics

- [Schritt 1: Erstellen von Lambda-Funktionen und einer DynamoDB-Tabelle](#)
- [Schritt 2: Erstellen Sie eine API WebSocket](#)
- [Schritt 3: Testen Ihrer API](#)
- [Schritt 4: Bereinigen](#)
- [Nächste Schritte: Automatisieren Sie mit CloudFormation](#)

Schritt 1: Erstellen von Lambda-Funktionen und einer DynamoDB-Tabelle

Laden Sie [die Vorlage zur App-Erstellung für CloudFormation](#) herunter und entpacken Sie sie. Sie verwenden diese Vorlage, um eine Amazon DynamoDB-Tabelle zu erstellen, in der der Client Ihrer App gespeichert wird. IDs Jeder verbundene Client hat eine eindeutige ID, die wir als Partitionsschlüssel der Tabelle verwenden werden. Diese Vorlage erstellt auch Lambda-Funktionen, die Ihre Client-Verbindungen in DynamoDB aktualisieren und das Senden von Nachrichten an verbundene Clients verarbeiten.

Um einen Stack zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **websocket-api-chat-app-tutorial** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres CloudFormation Stacks `CREATE_COMPLETE` lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen Sie eine API WebSocket

Sie erstellen eine WebSocket API, um Client-Verbindungen zu verarbeiten und Anfragen an die Lambda-Funktionen weiterzuleiten, die Sie in Schritt 1 erstellt haben.

Um eine API zu erstellen WebSocket

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) aus. Wählen Sie dann für WebSocket API die Option Build aus.
3. Geben Sie in API name (API-Name) **websocket-chat-app-tutorial** ein.
4. Wählen Sie als IP-Adresstyp die Option aus IPv4.
5. Geben Sie in Route selection expression (Ausdruck für die Routenauswahl) **request.body.action** ein. Der Ausdruck für die Routenauswahl legt die Route fest, die von API Gateway aufgerufen wird, wenn ein Client eine Nachricht sendet.
6. Wählen Sie Next (Weiter) aus.
7. Wählen Sie in Predefined routes (Vordefinierte Routen) die Optionen Add \$connect (\$connect hinzufügen), Add \$disconnect (\$disconnect hinzufügen) und Add \$default (\$default hinzufügen) aus. Die Routen \$connect und \$disconnect sind spezielle Routen, die API Gateway automatisch aufruft, wenn ein Client eine Verbindung zu einer API herstellt oder trennt. API Gateway ruft die Route \$default auf, wenn keine anderen Routen mit einer Anforderung übereinstimmen.
8. Wählen Sie in Custom routes (Benutzerdefinierte Routen) Add custom route (Benutzerdefinierte Route hinzufügen) aus. Geben Sie in Route key (Routenschlüssel) **sendMessage** ein. Diese benutzerdefinierte Route verarbeitet Nachrichten, die an verbundene Clients gesendet werden.
9. Wählen Sie Weiter aus.
10. Wählen Sie in Attach integrations (Integrationen anfügen) für jede Route und jeden Integration type (Integrationstyp) „Lambda“ aus.

Wählen Sie für Lambda die entsprechende Lambda-Funktion aus, mit der Sie CloudFormation in Schritt 1 erstellt haben. Der Name jeder Funktion entspricht einer Route. Für die Route \$connect

wählen Sie beispielsweise die Funktion mit dem Namen **websocket-chat-app-tutorial-ConnectHandler** aus.

11. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. Standardmäßig erstellt API Gateway den Phasennamen `production` und stellt Ihre API automatisch für diese Phase bereit. Wählen Sie Weiter aus.
12. Wählen Sie Create and deploy (Erstellen und bereitstellen) aus.

Schritt 3: Testen Ihrer API

Als Nächstes testen Sie Ihre API, um sicherzustellen, dass sie korrekt funktioniert. Stellen Sie mit dem Befehl `wscat` eine Verbindung zur API her.

So rufen Sie die URL für den Aufruf Ihrer API ab

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Stages (Phasen) und anschließend `production` (Produktion) aus.
4. Notieren Sie sich die URL Ihrer API. WebSocket Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.

So stellen Sie eine Verbindung zu Ihrer API her

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zu Ihrer API her. Wenn Sie die Verbindung zu Ihrer API herstellen, ruft API Gateway die Route `$connect` auf. Bei Aufruf dieser Route wird eine Lambda-Funktion aufgerufen, mit der die Verbindungs-ID in DynamoDB gespeichert wird.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Öffnen Sie ein neues Terminal. Führen Sie den Befehl `wscat` erneut mit den folgenden Parametern aus.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```



```
Connected (press CTRL+C to quit)
```

Anschließend erhalten Sie zwei verbundene Clients, die Nachrichten austauschen können.

So senden Sie eine Nachricht

- API Gateway legt basierend auf dem Ausdruck für die Routenauswahl Ihrer API fest, welche Route aufgerufen wird. Der Ausdruck für die Routenauswahl Ihrer API ist `$request.body.action`. Daher ruft API Gateway die Route `sendmessage` auf, wenn Sie die folgende Nachricht senden:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

Die Lambda-Funktion, die der aufgerufenen Route zugeordnet ist, sammelt den Client IDs von DynamoDB. Anschließend ruft die Funktion die API-Gateway-Management-API auf und sendet die Nachricht an diese Clients. Alle verbundenen Clients erhalten die folgende Nachricht:

```
< hello, everyone!
```

So rufen Sie die Route „`$default`“ Ihrer API auf

- API Gateway ruft die Standardroute Ihrer API auf, wenn ein Client eine Nachricht sendet, die nicht mit den von Ihnen definierten Routen übereinstimmt. Die mit der Route `$default` verknüpfte Lambda-Funktion verwendet die API-Gateway-Management-API, um die Client-Daten zur Verbindung zu senden.

```
test
```

Use the `sendmessage` route to send a message. Your info:

```
{"ConnectedAt":"2022-01-25T18:50:04.673Z","Identity":  
{"SourceIp":"192.0.2.1","UserAgent":null},"LastActiveAt":"2022-01-25T18:50:07.642Z","connec
```

So trennen Sie die Verbindung zu Ihrer API

- Zur Trennung der Verbindung zu Ihrer API drücken Sie **CTRL+C**. Wenn ein Client die Verbindung zu Ihrer API trennt, ruft API Gateway die Route `$disconnect` Ihrer API auf. Die Lambda-Integration für die Route `$disconnect` Ihrer API entfernt die Verbindungs-ID aus DynamoDB.

Schritt 4: Bereinigen

Um unnötige Kosten zu verhindern, löschen Sie die Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben. Mit den folgenden Schritten werden Ihr Stack und Ihre CloudFormation API gelöscht. WebSocket

Um eine WebSocket API zu löschen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIs-Seite Ihre API aus. `websocket-chat-app-tutorial` Wählen Sie Actions (Aktionen) und Delete (Löschen) aus. Bestätigen Sie anschließend Ihre Auswahl.

Um einen CloudFormation Stapel zu löschen

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

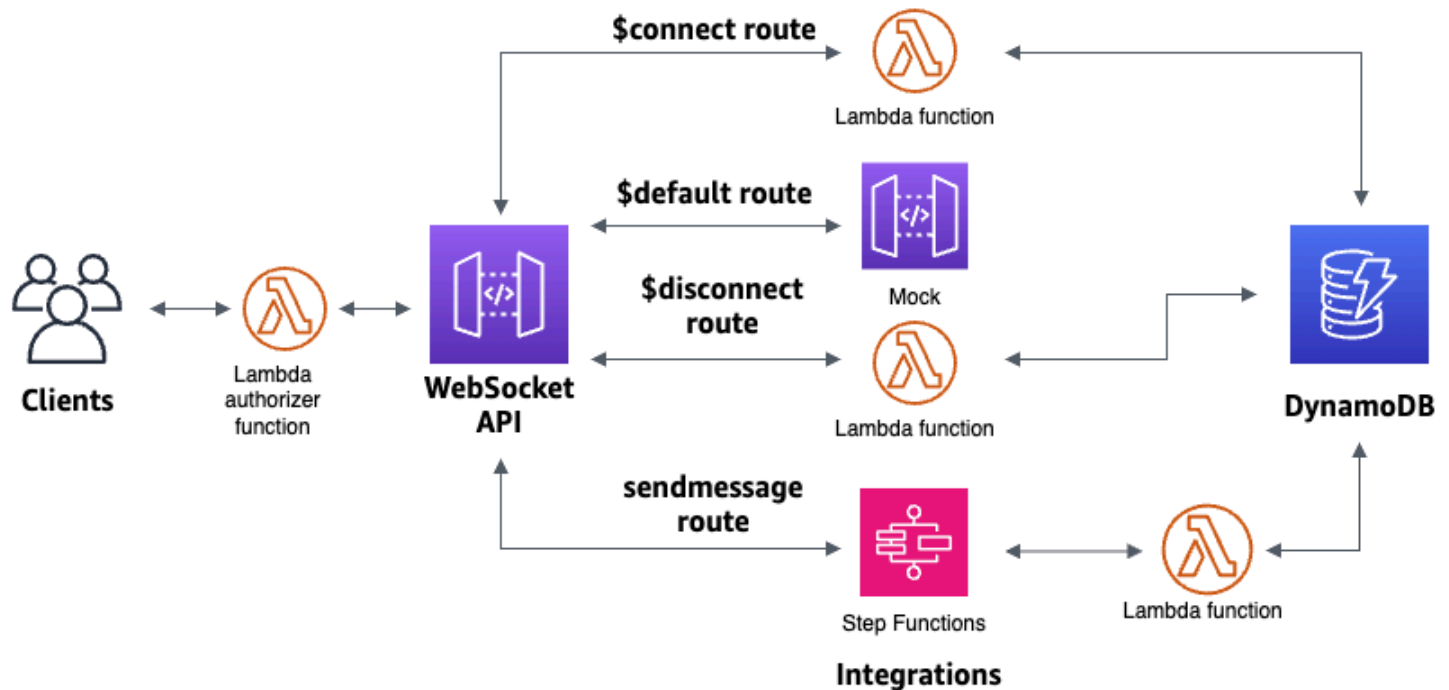
Nächste Schritte: Automatisieren Sie mit CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter [ws-chat-app.yaml](#).

Tutorial: Erstellen Sie eine WebSocket API mit einer AWS Integration

In diesem Tutorial erstellen Sie eine serverlose Broadcast-Anwendung mit einer WebSocket API. Kunden können Nachrichten erhalten, ohne Updates abfragen zu müssen.

In diesem Tutorial sehen Sie, wie Nachrichten an verbundene Clients übertragen werden. Enthalten sind außerdem ein Beispiel für einen Lambda-Genehmiger, eine Scheinintegration und eine Nicht-Proxy-Integration für Step Functions.



Nachdem Sie Ihre Ressourcen mithilfe einer CloudFormation Vorlage erstellt haben, verwenden Sie die API Gateway Gateway-Konsole, um eine WebSocket API zu erstellen, die sich in Ihre AWS Ressourcen integrieren lässt. Sie fügen Ihrer API einen Lambda-Authorizer hinzu und erstellen eine AWS Serviceintegration mit Step Functions, um eine State-Machine-Ausführung zu starten. Der Zustandsautomat für die Step Functions ruft eine Lambda-Funktion auf, die eine Nachricht an alle verbundenen Clients sendet.

Nachdem Sie Ihre API erstellt haben, testen Sie die Verbindung zu Ihrer API und prüfen, ob Nachrichten gesendet und empfangen werden. Dieses Tutorial nimmt ungefähr 30 Minuten in Anspruch.

Topics

- [Voraussetzungen](#)
- [Schritt 1: Erstellen von Ressourcen](#)
- [Schritt 2: Erstellen Sie eine API WebSocket](#)
- [Schritt 3: Erstellen eines Lambda-Genehmigers](#)
- [Schritt 4: Erstellen einer bidirektionalen Scheinintegration](#)
- [Schritt 5: Erstellen einer Nicht-Proxy-Integration mit Step Functions](#)

- [Schritt 6: Ihre API testen](#)
- [Schritt 7: Bereinigen](#)
- [Nächste Schritte](#)

Voraussetzungen

Sie benötigen die folgenden Voraussetzungen:

- Ein AWS Konto und ein AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter [Voraussetzungen](#).
- `wscat` für einen Verbindungsaufbau zu Ihrer API. Weitere Informationen finden Sie unter [the section called “Wird verwendetwscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).

Wir empfehlen, dass Sie das WebSocket Chat-App-Tutorial abschließen, bevor Sie mit diesem Tutorial beginnen. Informationen zum Ausfüllen des WebSocket Chat-App-Tutorials finden Sie unter [the section called “Tutorial: Erstellen Sie eine Chat-App WebSocket”](#).

Schritt 1: Erstellen von Ressourcen

Laden Sie [die Vorlage zur App-Erstellung für CloudFormation](#) herunter und entpacken Sie sie. Sie verwenden diese Vorlage, um Folgendes zu erstellen:

- Lambda-Funktionen, die API-Anforderungen bearbeiten und den Zugriff auf Ihre API genehmigen.
- Eine DynamoDB-Tabelle zum Speichern der Client IDs - und Hauptbenutzeridentifikation, die vom Lambda-Autorisierer zurückgegeben wurde.
- Einen Zustandsautomaten für die Step Functions zur Versendung von Nachrichten an verbundene Clients.

Um einen Stack zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.

6. Geben Sie für Stack-Name die Zeichenfolge **websocket-step-functions-tutorial** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wählen Sie die Registerkarte Ausgaben, um Ihre erstellten Ressourcen und deren Eigenschaften zu sehen ARNs. Wenn der Status Ihres CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen Sie eine API WebSocket

Sie erstellen eine WebSocket API, um Client-Verbindungen zu verarbeiten und Anfragen an die Ressourcen weiterzuleiten, die Sie in Schritt 1 erstellt haben.

Um eine WebSocket API zu erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) aus. Wählen Sie dann für WebSocket API die Option Build aus.
3. Geben Sie in API name (API-Name) **websocket-step-functions-tutorial** ein.
4. Wählen Sie als IP-Adresstyp die Option aus IPv4.
5. Geben Sie in Route selection expression (Ausdruck für die Routenauswahl) **request.body.action** ein.

Der Ausdruck für die Routenauswahl legt die Route fest, die von API Gateway aufgerufen wird, wenn ein Client eine Nachricht sendet.

6. Wählen Sie Weiter aus.
7. Wählen Sie unter Vordefinierte Routen die Optionen \$connect hinzufügen, \$disconnect hinzufügen und \$default hinzufügen aus.

Die Routen \$connect und \$disconnect sind spezielle Routen, die API Gateway automatisch aufruft, wenn ein Client eine Verbindung zu einer API herstellt oder trennt. API Gateway ruft

die \$default-Route auf, wenn keine anderen Routen mit einer Anforderung übereinstimmen. Nachdem Sie Ihre API erstellt haben, erstellen Sie eine benutzerdefinierte Route, um eine Verbindung zu den Step Functions herzustellen.

8. Wählen Sie Weiter aus.
9. Gehen Sie für Integration für \$connect wie folgt vor:
 - a. Wählen Sie unter Integrationstyp die Option Lambda aus.
 - b. Wählen Sie für die Lambda-Funktion die entsprechende \$connect Lambda-Funktion aus, mit der Sie CloudFormation in Schritt 1 erstellt haben. Der Lambda-Funktionsname sollte mit **websocket-step** beginnen.
10. Gehen Sie für Integration für \$disconnect wie folgt vor:
 - a. Wählen Sie unter Integrationstyp die Option Lambda aus.
 - b. Wählen Sie für die Lambda-Funktion die entsprechende \$disconnect Lambda-Funktion aus, mit der Sie CloudFormation in Schritt 1 erstellt haben. Der Lambda-Funktionsname sollte mit **websocket-step** beginnen.
11. Wählen Sie für Integration für \$default die Option Mock aus.

Bei einer Mock-Integration verwaltet API Gateway die Routenantwort ohne ein Integrations-Backend.

12. Wählen Sie Weiter aus.
13. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. Standardmäßig erstellt API Gateway eine Stufe namens Produktion und stellt Ihre API automatisch für diese Stufe bereit. Wählen Sie Weiter aus.
14. Wählen Sie Create and deploy (Erstellen und bereitstellen) aus.

Schritt 3: Erstellen eines Lambda-Genehmigers

Um den Zugriff auf Ihre WebSocket API zu kontrollieren, erstellen Sie einen Lambda-Autorisierer. Die CloudFormation Vorlage hat die Lambda-Autorisierungsfunktion für Sie erstellt. Sie können die Lambda-Funktion auch in der Lambda-Konsole sehen. Der Name muss mit **websocket-step-functions-tutorial-AuthorizerHandler** beginnen. Diese Lambda-Funktion verweigert alle WebSocket API-Aufrufe, es sei denn, der Authorization Header ist es. Allow Außerdem übergibt die Lambda-Funktion die \$context.authorizer.principalId-Variable auch an Ihre API, die später in der DynamoDB-Tabelle zur Identifizierung der API-Aufrufer verwendet wird.

In diesem Schritt konfigurieren Sie die \$connect-Route zur Verwendung des Lambda-Genehmigers.

Erstellen eines Lambda-Genehmigers

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Klicken Sie auf Genehmiger erstellen.
4. Geben Sie unter Genehmigername **LambdaAuthorizer** ein.
5. Geben Sie für Authorizer ARN den Namen des Autorisierers ein, der mit der CloudFormation Vorlage erstellt wurde. Der Name muss mit **websocket-step-functions-tutorial-AuthorizerHandler** beginnen.

Note

Wir empfehlen, diesen Beispielautorisierer nicht für Ihre Produktion zu verwenden. APIs

6. Wählen Sie unter Identitätsquellentyp die Option Header aus. Geben Sie für Key (Schlüssel) **Authorization** ein.
7. Wählen Sie Autorisierer erstellen.

Nachdem Sie Ihren Genehmiger erstellt haben, hängen Sie ihn an die \$connect-Route Ihrer API an.

So fügen Sie einen Genehmiger zu der \$connect-Route hinzu

1. Klicken Sie im Hauptnavigationsbereich auf Routen.
2. Wählen Sie die \$connect-Route aus.
3. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus.
4. Öffnen Sie das Dropdown-Menü Genehmigung und wählen Sie Ihren Anforderungsgenehmiger aus.
5. Wählen Sie Änderungen speichern aus.

Schritt 4: Erstellen einer bidirektionalen Scheinintegration

Als Nächstes erstellen Sie die bidirektionale Scheinintegration für die `$default-Route`. Mit einer Scheinintegration können Sie eine Antwort an den Client senden, ohne ein Backend zu verwenden. Wenn Sie eine Integration für die `$default-Route` erstellen, können Sie damit Ihren Kunden zeigen, wie sie mit Ihrer API interagieren können.

Sie konfigurieren die `$default-Route`, um Clients darüber zu informieren, dass sie die `SendMessage-Route` verwenden sollen.

So erstellen Sie eine Scheinintegration

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie die `$default-Route` und dann die Registerkarte Integrationsanforderung aus.
3. Klicken Sie unter Anforderungsvorlagen auf Bearbeiten.
4. Geben Sie unter Ausdruck für die Vorlagenauswahl **200** ein und klicken Sie dann auf Bearbeiten.
5. Wählen Sie auf der Registerkarte Integrationsanforderung unter Anforderungsvorlagen die Option Vorlage erstellen aus.
6. Geben Sie unter Vorlagenschlüssel **200** ein.
7. Geben Sie für Vorlagentext die folgende Zuweisungsvorlage ein:

```
{"statusCode": 200}
```

Wählen Sie Create template (Vorlage erstellen) aus.

Das Ergebnis sollte wie folgt aussehen:

Route request
Integration request
Integration response
Route response

Integration request settings

Integration type [Info](#)
Mock

Timeout
29000 ms

[Edit](#)

Request templates (1)

[Edit](#)
[Create template](#)

Use request templates to transform the incoming message before sending it to the integration. API Gateway uses a template selection expression to determine which template to use. Name the template with a key that matches the result of the selection expression.

Template selection expression
200

200

[Edit](#)
[Delete](#)

```

1  {"statusCode": 200}
2
3

```

8. Wählen Sie im Bereich `$default-Route` die Option Bidirektionale Kommunikation aktivieren aus.
9. Wählen Sie die Registerkarte Integrationsantwort und dann Integrationsantwort erstellen aus.
10. Geben Sie unter Antwortschlüssel **`$default`** ein.
11. Geben Sie unter Vorlagen-Auswahlausdruck **`200`** ein.
12. Wählen Sie Create response (Antwort erstellen) aus.
13. Klicken Sie unter Zuweisungsvorlagen auf Vorlage erstellen.
14. Geben Sie unter Vorlagenschlüssel **`200`** ein.
15. Geben Sie für Antwortvorlage die folgende Zuweisungsvorlage ein:

```

{"Use the sendMessage route to send a message. Connection ID:
 $context.connectionId"}

```

16. Wählen Sie Create template (Vorlage erstellen) aus.

Das Ergebnis sollte wie folgt aussehen:

Route request

Integration request

Integration response

Route response

Integration response settings

[Create integration response](#)

Integration responses allow you to configure transformations on the outgoing message's payload using response template definitions. The response chosen is based on the response key found in the outgoing message after evaluating the response selection expression.

\$default[Edit](#)[Delete](#)**Template selection expression**

200

Response templates

[Create template](#)**200**[Edit](#)[Delete](#)

```
1  {"Use the sendmessage route to send a message. Connection ID:
   $context.connectionId"}
2
3
```

Schritt 5: Erstellen einer Nicht-Proxy-Integration mit Step Functions

Als Nächstes erstellen Sie eine sendmessage-Route. Kunden können die sendmessage-Route zur Versendung einer Nachricht an alle verbundenen Clients aufrufen. Die SendMessage-Route hat eine Nicht-Proxy-Serviceintegration mit AWS Step Functions. Die Integration ruft den [StartExecution](#)-Befehl für den Step Functions Functions-Zustandsmaschine auf, den die CloudFormation Vorlage für Sie erstellt hat.

So erstellen Sie eine Nicht-Proxy-Integration

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie in Route key (Routenschlüssel) **sendMessage** ein.
4. Wählen Sie unter Integrationstyp die Option AWS -Service aus.
5. Geben Sie AWS unter Region die Region ein, in der Sie Ihre Vorlage bereitgestellt haben.
CloudFormation
6. Wählen Sie unter AWS -Service die Option Step Functions aus.
7. Wählen Sie in HTTP method POST.
8. Für Aktionsname geben Sie **StartExecution** ein.
9. Geben Sie für Ausführungsrolle die Ausführungsrolle ein, die durch die CloudFormation Vorlage erstellt wurde. Der Name sollte sein WebSocketTutorialApiRole.
10. Wählen Sie Create route (Route erstellen) aus.

Als Nächstes erstellen Sie eine Zuweisungsvorlage, um Anforderungsparameter an die Zustandsmaschine für Step Functions zu senden.

Erstellen einer Zuweisungsvorlage

1. Wählen Sie die sendMessage-Route und dann die Registerkarte Integrationsanforderung aus.
2. Klicken Sie im Abschnitt Vorlagenanforderung auf Bearbeiten.
3. Geben Sie unter Vorlagen-Auswahlausdruck **\\$default** ein.
4. Wählen Sie Bearbeiten aus.
5. Klicken Sie im Abschnitt Vorlagenanforderung auf Vorlage erstellen.
6. Geben Sie unter Vorlagenschlüssel **\\$default** ein.
7. Geben Sie für Vorlagentext die folgende Zuweisungsvorlage ein:

```
#set($domain = "$context.domainName")
#set($stage = "$context.stage")
#set($body = $input.json('$'))
#set($getMessage = $util.parseJson($body))
#set($mymessage = $getMessage.message)
{
```

```
"input": "{\\"domain\\": \\"$domain\\", \\"stage\\": \\"$stage\\", \\"message\\": \\"$mymessage\\"}",  
"stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:WebSocket-Tutorial-StateMachine"  
}
```

Ersetzen Sie das *stateMachineArn* durch den ARN der Zustandsmaschine, die von erstellt wurde CloudFormation.

Die Zuweisungsvorlage führt Folgendes aus:

- Erzeugt die Variable `$domain` mithilfe der Kontextvariable `domainName`.
- Erzeugt die Variable `$stage` mithilfe der Kontextvariable `stage`.

Die Variablen `$domain` und `$stage` sind für die Erzeugung einer Callback-URL erforderlich.

- Extrahiert die `message`-Eigenschaft aus der eingehenden `sendmessage`-JSON-Nachricht.
- Erzeugt die Eingabe für den Zustandsautomaten. Die Eingabe ist die Domäne und die Phase der WebSocket API sowie die Nachricht von der `sendmessage` Route.

8. Wählen Sie `Create template` (Vorlage erstellen) aus.

Request templates (1)

[Edit](#)[Create template](#)

Use request templates to transform the incoming message before sending it to the integration. API Gateway uses a template selection expression to determine which template to use. Name the template with a key that matches the result of the selection expression.

Template selection expression

\\$default

\\$default

[Edit](#)[Delete](#)

```
1  #set($domain = "$context.domainName")
2  #set($stage = "$context.stage")
3  #set($body = $input.json('$'))
4  #set($getMessage = $util.parseJson($body))
5  #set($mymessage = $getMessage.message)
6  {
7    "input": "{\"domain\": \"$domain\", \"stage\": \"$stage\", \"message\": \"$mymessage\"}",
8    "stateMachineArn": "arn:aws:states:us-east-2:111222333:stateMachine:WebSocket-Tutorial-StateMachine"
9  }
```

Sie können eine Nicht-Proxy-Integration auf den Routen \$connect oder \$disconnect erstellen, um direkt eine Verbindungs-ID aus der DynamoDB-Tabelle hinzuzufügen oder zu entfernen, ohne eine Lambda-Funktion aufzurufen.

Schritt 6: Ihre API testen

Als Nächstes veröffentlichen und testen Sie Ihre API, um sicherzustellen, dass sie korrekt funktioniert. Sie verwenden den `wscat` Befehl, um eine Verbindung zur API herzustellen, und dann verwenden Sie einen `Slash`-Befehl, um einen Ping-Frame zu senden, um die Verbindung zur WebSocket API zu überprüfen.

Stellen Sie Ihre API bereit

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Routen.
3. Klicken Sie auf Deploy API.
4. Wählen Sie unter Stufe Produktion aus.
5. (Optional) Geben Sie unter Bereitstellungsbeschreibung eine Beschreibung ein.
6. Wählen Sie Bereitstellen.

Nachdem Sie Ihre API bereitgestellt haben, können Sie sie aufrufen. Verwenden Sie die Aufruf-URL, um Ihre API aufzurufen.

So rufen Sie die URL für den Aufruf Ihrer API auf

1. Wählen Sie Ihre API aus.
2. Wählen Sie Stages (Phasen) und anschließend production (Produktion) aus.
3. Notieren Sie sich die URL Ihrer API. WebSocket Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.

Jetzt, da Sie Ihre Aufruf-URL haben, können Sie die Verbindung zu Ihrer WebSocket API testen.

So testen Sie die Verbindung zu Ihrer API

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zu Ihrer API her. Zuerst testen Sie die Verbindung, indem Sie den /ping-Pfad aufrufen.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H  
"Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Geben Sie den folgenden Befehl ein, um den Kontrollrahmen anzupingen. Sie können einen Kontrollrahmen für Keepalive-Zwecke von der Client-Seite aus verwenden.

```
/ping
```

Das Ergebnis sollte wie folgt aussehen:

```
< Received pong (data: "")
```

Jetzt, da Sie die Verbindung getestet haben, können Sie Ihre API auf ordnungsgemäße Funktion prüfen. In diesem Schritt öffnen Sie ein neues Terminalfenster, damit die WebSocket API eine Nachricht an alle verbundenen Clients senden kann.

So testen Sie Ihre API

1. Öffnen Sie ein neues Terminal. Führen Sie den Befehl `wscat` erneut mit den folgenden Parametern aus.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H  
"Authorization: Allow"
```

```
Connected (press CTRL+C to quit)
```

2. API Gateway legt basierend auf dem Ausdruck für die Routenanforderungsauswahl Ihrer API fest, welche Route aufgerufen wird. Der Ausdruck für die Routenauswahl Ihrer API lautet `$request.body.action`. Daher ruft API Gateway die Route `sendmessage` auf, wenn Sie die folgende Nachricht senden:

```
{"action": "sendmessage", "message": "hello, from Step Functions!"}
```

Die der Route zugeordnete Zustandsmaschine für Step Functions ruft eine Lambda-Funktion mit der Nachricht und der Callback-URL auf. Die Lambda-Funktion ruft die API-Gateway-Management-API auf und sendet die Nachricht an alle verbundenen Clients. Alle Clients erhalten die folgende Nachricht:

```
< hello, from Step Functions!
```

Nachdem Sie Ihre WebSocket API getestet haben, können Sie die Verbindung zu Ihrer API trennen.

So trennen Sie die Verbindung zu Ihrer API

- Zur Trennung der Verbindung zu Ihrer API drücken Sie CTRL+C.

Wenn ein Client die Verbindung zu Ihrer API trennt, ruft API Gateway die Route `$disconnect` Ihrer API auf. Die Lambda-Integration der `$disconnect`-Route Ihrer API entfernt die Verbindungs-ID aus DynamoDB.

Schritt 7: Bereinigen

Um unnötige Kosten zu verhindern, löschen Sie die Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben. Mit den folgenden Schritten werden Ihr CloudFormation Stack und Ihre WebSocket API gelöscht.

Um eine WebSocket API zu löschen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie auf der APIsSeite Ihre Websocket-API aus.
3. Wählen Sie Actions (Aktionen) und Delete (Löschen) aus. Bestätigen Sie anschließend Ihre Auswahl.

Um einen Stapel zu löschen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Nächste Schritte

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. [Ein Beispiel für eine CloudFormation Vorlage, die diese Aktionen für dieses Tutorial automatisiert, finden Sie unter ws-sfn.zip.](#)

API-Gateway-REST-APIs

Eine REST-API in API Gateway ist eine Sammlung von Ressourcen und Methoden, die mit Backend-HTTP-Endpunkten, Lambda-Funktionen oder anderen AWS-Services integriert sind. Sie können API Gateway-Funktionen verwenden, um Ihnen bei allen Aspekten des API-Lebenszyklus zu helfen, von der Erstellung bis zur Überwachung Ihrer Produktions-APIs.

API Gateway-REST-APIs verwenden ein Anfrage-/Antwortmodell, bei dem ein Client eine Anfrage an einen Service sendet und der Service synchron zurückantwortet. Diese Art von Modell eignet sich für viele verschiedene Arten von Anwendungen, die von synchroner Kommunikation abhängen.

Themen

- [Entwickeln einer REST-API in API Gateway](#)
- [Veröffentlichen Sie REST, APIs damit Kunden es aufrufen können](#)
- [Optimieren Sie die Leistung von REST APIs](#)
- [Verteilen Sie Ihren REST APIs an Kunden in API Gateway](#)
- [Schützen Sie Ihren REST APIs in API Gateway](#)
- [REST APIs in API Gateway überwachen](#)

Entwickeln einer REST-API in API Gateway

In Amazon API Gateway erstellen Sie eine REST-API als eine Sammlung programmierbarer Entitäten, die als API Gateway-[Ressourcen](#) bekannt ist. Sie verwenden beispielsweise eine [RestApi](#)Ressource, um eine API darzustellen, die eine Sammlung von [Ressourcenentitäten](#) enthalten kann.

Jede Resource-Entität kann über eine oder mehrere [Methoden](#)-Ressourcen verfügen.

Eine Method ist eine eingehende Anfrage, die vom Client übermittelt wird, und kann die folgenden Anfrageparameter enthalten: einen Pfadparameter, einen Header oder einen Abfragezeichenfolgenparameter. Zusätzlich kann die Anfrage, abhängig von der HTTP-Methode, einen Body enthalten. Die Methode definiert, wie der Client auf die exponierte Resource zugreift. Erstellen Sie eine Ressource unter [Integration](#), um die Method mit einem Backend-Endpunkt (auch als Integrationsendpunkt bekannt) zu integrieren. Dadurch wird die eingehende Anforderung an eine bestimmte Integrationsendpunkt-URI weitergeleitet. Falls erforderlich, können Sie Anfrageparameter oder den Anfrage-Body transformieren, um die Anforderungen des Backends zu erfüllen, oder Sie

erstellen eine Proxyintegration, bei der API Gateway die gesamte Anfrage in einem standardisierten Format an die Integrationsendpunkt-URI sendet und anschließend die Antwort direkt an den Client weiterleitet.

Für Antworten können Sie eine [MethodResponse](#)-Ressource erstellen, die eine vom Client empfangene Antwort darstellt, und Sie können eine [IntegrationResponse](#)-Ressource erstellen, die die Antwort darstellt, die vom Backend zurückgegeben wird. Verwenden Sie eine Integrationsantwort, um die Backend-Antwortdaten zu transformieren, bevor die Daten an den Client zurückgegeben werden, oder um die Backend-Antwort unverändert an den Client zu übermitteln.

Beispielressource für eine REST-API

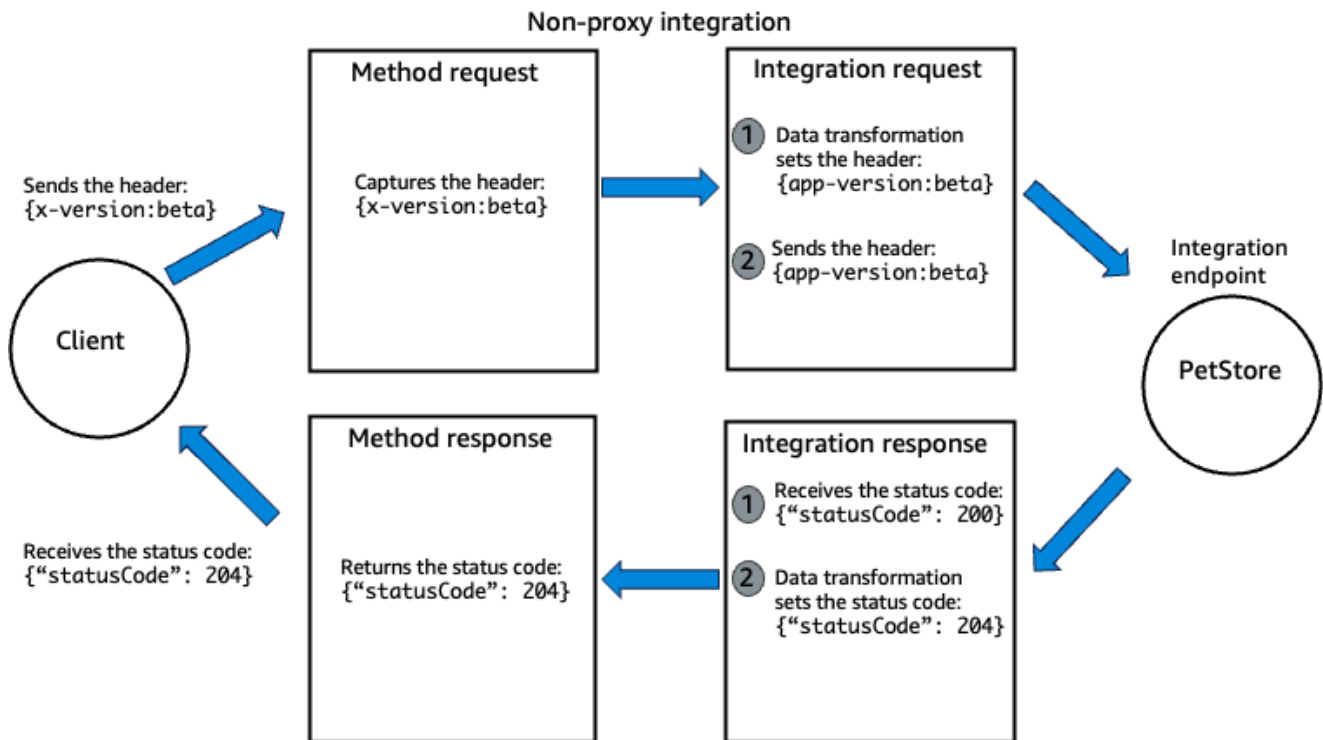
Das folgende Diagramm zeigt, wie API Gateway dieses request/response Modell für einen HTTP-Proxy und eine HTTP-Nicht-Proxy-Integration für die GET /pets Ressource implementiert. Der Client sendet den Header `x-version:beta` an API Gateway und API Gateway sendet den Statuscode 204 an den Client.

In der Nicht-Proxy-Integration führt API Gateway Datentransformationen durch, um die Backend-Anforderungen zu erfüllen, indem die Integrationsanfrage und die Integrationsantwort angepasst werden. Bei einer Nicht-Proxy-Integration können Sie auf den Body in der Methodenanforderung zugreifen, jedoch transformieren Sie ihn in der Integrationsanforderung. Wenn der Integrationsendpunkt eine Antwort mit einem Body zurückgibt, greifen Sie in der Integrationsantwort darauf zu und transformieren ihn. Sie können den Body in der Methodenantwort nicht ändern.

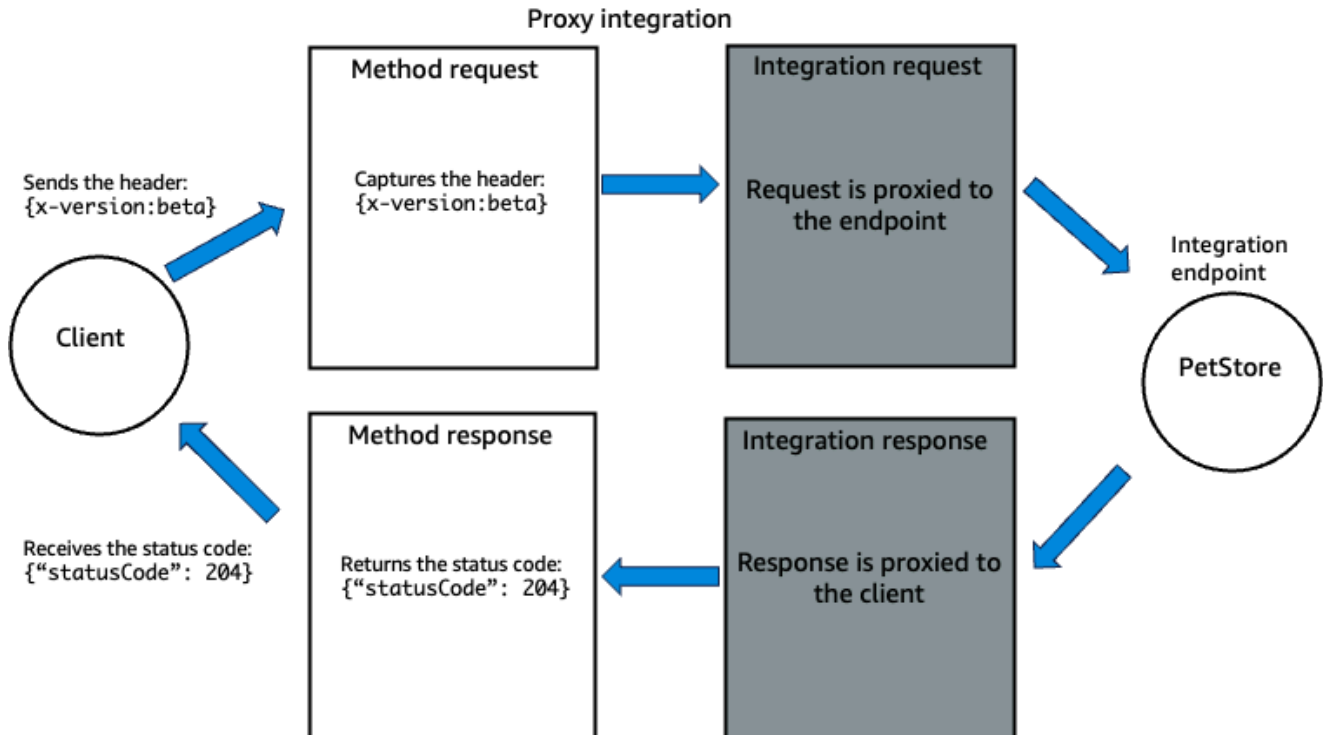
Bei der Proxyintegration ändert der Integrationsendpunkt die Anfrage und Antwort. API Gateway verändert weder die Integrationsanfrage noch die Integrationsantwort und sendet die eingehende Anfrage unverändert an das Backend.

Unabhängig vom Integrationstyp hat der Client eine Anfrage an API Gateway gesendet und API Gateway hat synchron geantwortet.

Non-proxy integration



Proxy integration



Die folgenden Beispielausführungsprotokolle zeigen, was API Gateway im vorherigen Beispiel protokollieren würde. Zur besseren Übersicht wurden einige Werte und erste Protokolle entfernt:

Non-proxy integration

```
Wed Feb 12 23:56:44 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:56:44 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:56:44 UTC 2025 : Method request path: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request query string: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request headers: {app-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:56:44 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:45 UTC 2025 : Received response. Status: 200, Integration latency:
123 ms
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
23:56:45 GMT}
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:56:45 UTC 2025 : Method response body after transformations: (null)
Wed Feb 12 23:56:45 UTC 2025 : Method response headers: {X-Amzn-Trace-Id=Root=1-
abcd-12345}
Wed Feb 12 23:56:45 UTC 2025 : Successfully completed execution
Wed Feb 12 23:56:45 UTC 2025 : Method completed with status: 204
```

Proxy integration

```
Wed Feb 12 23:59:42 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:59:42 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:59:42 UTC 2025 : Method request path: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request query string: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request headers: { x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:59:42 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
```

```
Wed Feb 12 23:59:43 UTC 2025 : Received response. Status: 204, Integration latency:
123 ms
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response body after transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response headers: {Date=Wed, 12 Feb 2025
23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Successfully completed execution
Wed Feb 12 23:59:43 UTC 2025 : Method completed with status: 204
```

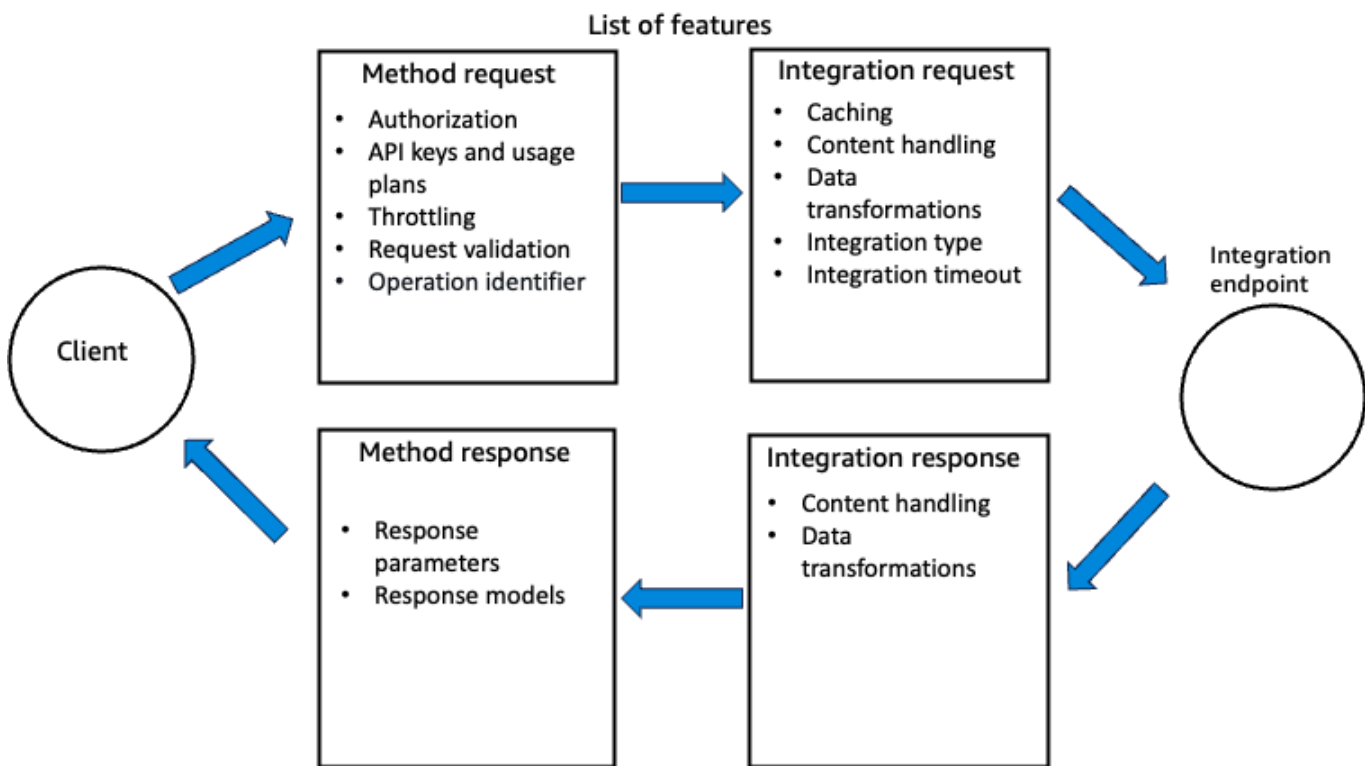
Informationen zum Importieren einer ähnlichen API und zum Testen in der AWS-Managementkonsole finden Sie in der [Beispiel-API](#).

Zusätzliche REST-API-Features für die Entwicklung

API Gateway unterstützt zusätzliche Features für die Entwicklung Ihrer REST-API. Zum Beispiel können Sie, um Ihren Kunden das Verständnis Ihrer API zu erleichtern, Dokumentation für die API bereitstellen. Um dies zu erreichen, fügen Sie eine [DocumentationPart](#)-Ressource für eine unterstützte API-Entität hinzu.

Um zu steuern, wie Clients eine API aufrufen, verwenden Sie [IAM-Berechtigungen](#), einen [Lambda-Genehmiger](#) oder einen [Amazon Cognito-Benutzerpool](#). Um die Nutzung Ihrer API zu messen, richten Sie [Nutzungspläne](#) ein, um die API-Anforderungen zu drosseln. Diese aktivieren Sie beim Erstellen oder Aktualisieren der API.

Das folgende Diagramm zeigt, welche Funktionen für die REST-API-Entwicklung verfügbar sind und wo im request/response Modell diese Funktionen konfiguriert sind.



Eine Einführung in die Erstellung einer API finden Sie unter [the section called “Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration”](#). Weitere Informationen zu den Funktionen von API Gateway, die Sie bei der Entwicklung einer REST-API verwenden können, finden Sie in den nachfolgenden Themen. Diese Themen enthalten konzeptionelle Informationen und Verfahren, die Sie mit der API Gateway Gateway-Konsole, der API Gateway Gateway-REST-API AWS CLI, dem oder einem der AWS SDKs ausführen können.

Themen

- [API-Endpunkttypen für REST APIs in API Gateway](#)
- [Sicherheitsrichtlinien für REST-APIs in API Gateway](#)
- [IP-Adresstypen für REST-APIs in API Gateway](#)
- [Methoden für REST APIs in API Gateway](#)
- [Steuern und verwalten Sie den Zugriff auf REST APIs in API Gateway](#)
- [Integrationen für REST APIs in API Gateway](#)
- [Validierung für REST APIs in API Gateway anfordern](#)
- [Datentransformationen für REST-APIs in API Gateway](#)
- [Gateway-Antworten für REST APIs in API Gateway](#)

- [CORS für REST APIs im API Gateway](#)
- [Binäre Medientypen für REST APIs in API Gateway](#)
- [Aufrufen von REST-APIs in API Gateway](#)
- [REST-APIs mit OpenAPI in API Gateway erstellen](#)

API-Endpunkttypen für REST APIs in API Gateway

Ein [API-Endpunkt](#)-Typ bezieht sich auf den Hostnamen der API. Der API-Endpunkttyp kann Edge-optimiert, Regional oder Privat sein, je nachdem, woher der Großteil des API-Datenverkehrs stammt.

Edge-optimierte API-Endpunkte

Ein [Edge-optimierter API-Endpunkt](#) leitet Anfragen in der Regel an den nächstgelegenen CloudFront Point of Presence (POP) weiter, was in Fällen hilfreich sein kann, in denen Ihre Kunden geografisch verteilt sind. Dies ist der Standard-Endpunkttyp für API Gateway REST APIs.

Edge-optimiert schreibt APIs die Namen von [HTTP-Headern groß \(z. B.\)](#). Cookie

CloudFront sortiert HTTP-Cookies in natürlicher Reihenfolge nach dem Namen des Cookies, bevor die Anfrage an Ihren Ursprung weitergeleitet wird. Weitere Informationen zur Art und Weise, wie Cookies CloudFront verarbeitet werden, finden Sie unter [Zwischenspeichern von Inhalten auf der Grundlage von Cookies](#).

Jeder von Ihnen für eine Edge-optimierte API verwendeter benutzerdefinierter Domänenname gilt für alle Regionen.

Regionale API-Endpunkte

Ein [regionaler API-Endpunkt](#) ist für Clients in derselben Region konzipiert. Wenn ein Client, der auf einer EC2-Instance ausgeführt wird, eine API in der gleichen Region aufruft oder wenn eine API eine kleine Anzahl von Clients mit hohen Anforderungen bedienen soll, reduziert eine regionale API den Verbindungs-Overhead.

Für eine regionale API gilt jeder von Ihnen verwendeter benutzerdefinierter Domainname spezifisch für die Region, in der die API bereitgestellt wird. Wenn Sie eine regionale API in mehreren Regionen bereitstellen, kann sie in allen Regionen den gleichen Domainnamen haben. Sie können benutzerdefinierte Domänen zusammen mit Amazon Route 53 verwenden, um Aufgaben wie [latenzbasiertes Routing](#) auszuführen. Weitere Informationen erhalten Sie unter [the section called](#)

[“Einrichten eines regionalen benutzerdefinierten Domainnamens”](#) und [the section called “Einrichten eines Edge-optimierten benutzerdefinierten Domainnamens in API Gateway”](#).

Regionale API-Endpunkte übergeben alle Headernamen unverändert.

 Note

In Fällen, in denen API-Clients geografisch verstreut sind, kann es dennoch sinnvoll sein, einen regionalen API-Endpunkt zusammen mit Ihrer eigenen CloudFront Amazon-Distribution zu verwenden, um sicherzustellen, dass API Gateway die API nicht mit CloudFront servicegesteuerten Distributionen verknüpft. Weitere Informationen zu diesem Anwendungsfall finden Sie unter [Wie richte ich API Gateway mit meiner eigenen CloudFront Distribution ein?](#) .

Private API-Endpunkte

Ein [privater API-Endpunkt](#) ist ein API-Endpunkt, auf den nur von Ihrer Amazon Virtual Private Cloud (VPC) aus zugegriffen werden kann, und zwar über einen Interface-VPC-Endpunkt, bei dem es sich um eine Endpunkt-Netzwerkschnittstelle (ENI) handelt, die Sie in Ihrer VPC erstellen. Weitere Informationen finden Sie unter [the section called “Privates REST APIs”](#).

Private API-Endpunkte übergeben alle Headernamen unverändert.

Öffentlichen oder privaten API-Endpunkttyp in API Gateway ändern

Das Ändern eines API-Endpunkttyps erfordert, dass Sie die Konfiguration der API aktualisieren. Sie können einen vorhandenen API-Typ mit der API Gateway-Konsole AWS CLI, dem oder einem AWS SDK für API Gateway ändern. Der Endpunkttyp kann erst wieder geändert werden, wenn die aktuelle Änderung abgeschlossen ist, aber Ihre API ist verfügbar.

Die folgenden Änderungen am Endpunkttyp werden nicht unterstützt:

- Von Edge-optimiert in regional oder privat
- Von regional in Edge-optimiert oder privat
- Von privat in regional

Sie können eine private API nicht in eine Edge-optimierte API ändern.

Wenn Sie eine öffentliche API von Edge-optimiert in regional (oder umgekehrt) umwandeln, sollten Sie beachten, dass eine Edge-optimierte API sich anders als eine regionale API verhalten kann. Bei einer Edge-optimierten API wird beispielsweise der Content-MD5-Header entfernt. Jeder MD5 Hashwert, der an das Backend übergeben wird, kann in einem Anforderungszeichenfolgenparameter oder einer Body-Eigenschaft ausgedrückt werden. Die regionale API übergibt diesen Header jedoch, obwohl dadurch der Header-Name einem anderen Namen zugeordnet werden kann. Sie müssen die Unterschiede verstehen, um zu entscheiden, wie Sie eine Edge-optimierte API zu einer regionalen oder umgekehrt machen.

Themen

- [API-Endpunkttyp über die API Gateway-Konsole ändern](#)
- [Verwenden Sie den AWS CLI , um den Typ eines API-Endpunkts zu ändern](#)

API-Endpunkttyp über die API Gateway-Konsole ändern

Um den API-Endpunkttyp Ihrer API zu ändern, führen Sie einen der folgenden Schritte aus:

So wandeln Sie einen öffentlichen Endpunkt von regional zu Edge-optimiert oder umgekehrt um

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie API-Einstellungen aus.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Wählen Sie als API-Endpunkttyp entweder Edge-optimiert oder Regional aus.
6. Wählen Sie Änderungen speichern aus.
7. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

So konvertieren Sie einen privaten Endpunkt in einen regionalen Endpunkt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Bearbeiten Sie die Ressourcenrichtlinie für Ihre API, um jegliche Erwähnung von VPCs VPC-Endpunkten zu entfernen, sodass API-Aufrufe sowohl von außerhalb Ihrer VPC als auch innerhalb Ihrer VPC erfolgreich sind.

4. Wählen Sie API-Einstellungen aus.
5. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
6. Wählen Sie für API-Endpunkttyp die Option Regional aus.
7. Wählen Sie Änderungen speichern aus.
8. Entfernen Sie die Ressourcenrichtlinie aus Ihrer API.
9. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

Da Sie den Endpunkttyp von privat zu regional migrieren, ändert API Gateway den IP-Adresstyp in IPv4. Weitere Informationen finden Sie unter [the section called “IP-Adresstypen für REST-APIs in API Gateway”](#).

So konvertieren Sie einen privaten Endpunkt in einen regionalen Endpunkt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Erstellen Sie eine Ressourcenrichtlinie, die Zugriff auf Ihre VPC oder Ihren VPC-Endpunkt gewährt. Weitere Informationen finden Sie unter [???](#).
4. Wählen Sie API-Einstellungen aus.
5. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
6. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
7. (Optional) Wählen Sie für VPC-Endpunkt IDs den VPC-Endpunkt aus IDs , den Sie mit Ihrer privaten API verknüpfen möchten.
8. Wählen Sie Änderungen speichern aus.
9. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

Da Sie den Endpunkttyp von „Regional“ zu „Privat“ migrieren, ändert API Gateway den IP-Adresstyp in „Dualstack“. Weitere Informationen finden Sie unter [the section called “IP-Adresstypen für REST-APIs in API Gateway”](#).

Verwenden Sie den AWS CLI , um den Typ eines API-Endpunkts zu ändern

Mit dem folgenden [update-rest-api](#) Befehl wird eine Edge-optimierte API auf eine regionale API aktualisiert:

```
aws apigateway update-rest-api \  
  --rest-api-id a1b2c3 \  
  --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

Die erfolgreiche Antwort verfügt über einen Statuscode von 200 OK und eine Nutzlast, die etwa wie folgt aussieht:

```
{  
  "createdDate": "2017-10-16T04:09:31Z",  
  "description": "Your first API with Amazon API Gateway. This is a sample API that  
integrates via HTTP with our demo Pet Store endpoints",  
  "endpointConfiguration": {  
    "types": "REGIONAL"  
  },  
  "id": "a1b2c3",  
  "name": "PetStore imported as edge-optimized"  
}
```

Mit dem folgenden [update-rest-api](#) Befehl wird eine regionale API auf eine Edge-optimierte API aktualisiert:

```
aws apigateway update-rest-api \  
  --rest-api-id a1b2c3 \  
  --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Da er [put-rest-api](#) für die Aktualisierung von API-Definitionen bestimmt ist, gilt er nicht für die Aktualisierung eines API-Endpunkttyps.

Sicherheitsrichtlinien für REST-APIs in API Gateway

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus einer minimalen TLS-Version und Verschlüsselungssuites, die von API Gateway angeboten werden. Wenn Ihre Clients einen TLS-Handshake mit Ihrer API oder Ihrem benutzerdefinierten Domain-Namen erfordern, setzt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite durch, die von API Gateway akzeptiert werden. Sicherheitsrichtlinien schützen Ihre APIs und benutzerdefinierten Domain-Namen vor Netzwerksicherheitsproblemen wie Manipulationen und Eavesdropping zwischen Client und Server.

API Gateway unterstützt ältere Sicherheitsrichtlinien und erweiterte Sicherheitsrichtlinien. TLS_1_0 und TLS_1_2 sind veraltete Sicherheitsrichtlinien. Verwenden Sie diese

Sicherheitsrichtlinien aus Gründen der Abwärtskompatibilität. Jede Richtlinie, die mit `SecurityPolicy_` beginnt, ist eine erweiterte Sicherheitsrichtlinie. Sie können diese Richtlinien für regulierte Workloads, erweiterte Governance oder die Verwendung von Post-Quanten-Kryptographie verwenden. Wenn Sie eine optimierte Sicherheitsrichtlinie verwenden, müssen Sie auch den Endpunktzugriffsmodus festlegen, um zusätzliche Governance bereitzustellen. Weitere Informationen finden Sie unter [the section called “Endpunktzugriffsmodus”](#).

Wie API Gateway Sicherheitsrichtlinien anwendet

Das folgende Beispiel zeigt am Beispiel der Sicherheitsrichtlinie, wie API Gateway `SecurityPolicy_TLS13_1_3_2025_09` Sicherheitsrichtlinien anwendet.

Die `SecurityPolicy_TLS13_1_3_2025_09` Sicherheitsrichtlinie akzeptiert TLS 1.3-Verkehr und lehnt TLS 1.2- und TLS 1.0-Verkehr ab. Für TLS 1.3-Verkehr akzeptiert die Sicherheitsrichtlinie die folgenden Verschlüsselungssammlungen:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway akzeptiert keine anderen Cipher Suites. Zum Beispiel würde die Sicherheitsrichtlinie jeglichen TLS 1.3-Verkehr ablehnen, der die AES128-SHA Cipher Suite verwendet. Weitere Informationen zu den unterstützten TLS-Versionen und Chiffren finden Sie unter [the section called “Unterstützte Sicherheitsrichtlinien”](#)

Um zu überwachen, welches TLS-Protokoll und welche Chiffren Clients für den Zugriff auf Ihr API Gateway verwendet haben, können Sie die `$context.cipherSuite` Kontextvariablen `$context.tlsVersion` und in Ihren Zugriffsprotokollen verwenden. Weitere Informationen finden Sie unter [the section called “Überwachen”](#).

Endpunktzugriffsmodus

Der Endpunktzugriffsmodus ist ein zusätzlicher Parameter, den Sie für jede REST-API oder benutzerdefinierten Domain-Namen angeben müssen, der eine mit `SecurityPolicy_` beginnende optimierte Sicherheitsrichtlinie verwendet. Sie tun dies, wenn Sie Ihre Ressource erstellen oder wenn Sie die Sicherheitsrichtlinie von einer älteren Richtlinie in eine erweiterte Richtlinie ändern.

Wenn der Endpunktzugriffsmodus auf eingestellt ist `STRICT`, müssen alle Anfragen an Ihre REST-API oder Ihren benutzerdefinierten Domainnamen die folgenden Prüfungen bestehen:

- Die Anfrage muss vom gleichen API-Gateway-Endpunkttyp stammen, von dem auch Ihre Ressource stammt. Dabei kann es sich um einen regionalen, einen Edge-optimierten oder einen privaten Endpunkt handeln.
- Wenn Sie einen regionalen oder privaten Endpunkt verwenden, verwendet API Gateway den SNI-Hostabgleich. Wenn Sie einen Edge-optimierten Endpunkt verwenden, entspricht API Gateway dem CloudFront Domain-Front-Schutz. [Weitere Informationen finden Sie unter Domain-Fronting.](#)

Wenn eine dieser Bedingungen nicht erfüllt ist, lehnt API Gateway die Anfrage ab. Wir empfehlen, wenn möglich den STRICT Endpunktzugriffsmodus zu verwenden.

Um einen vorhandenen API- oder Domainnamen zur Verwendung des strikten Endpunktzugriffsmodus zu migrieren, aktualisieren Sie zunächst Ihre Sicherheitsrichtlinie auf eine erweiterte Sicherheitsrichtlinie und lassen Sie den Endpunktzugriffsmodus auf eingestellt BASIC. Nachdem Sie Ihre Datenverkehrs- und Zugriffsprotokolle überprüft haben, stellen Sie den Endpunktzugriffsmodus auf ein STRICT. Wenn Sie den Endpunktzugriffsmodus von STRICT zu migrieren BASIC, ist Ihr Endpunkt etwa 15 Minuten lang nicht verfügbar, da die Änderungen übernommen werden.

Sie sollten den Endpunktzugriffsmodus STRICT für bestimmte Anwendungsarchitekturen nicht auf einstellen, sondern den Endpunktzugriffsmodus auf BASIC. Die folgende Tabelle zeigt einige Anwendungsarchitekturen und eine Empfehlung, damit Ihre REST-API oder Ihr benutzerdefinierter Domainname den STRICT Endpunktzugriffsmodus verwenden kann.

Architektur	Vorgeschlagene Migration
Verwenden eines VPC-Endpunkts für den Zugriff auf einen öffentlichen benutzerdefinierten Domainnamen.	Diese Architektur verwendet endpunkübergreifenden Datenverkehr. Wir empfehlen Ihnen, zu the section called “Benutzerdefinierte Domainnamen für private APIs” zu migrieren.
Verwenden Sie eine beliebige Methode, um eine private API aufzurufen, die keinen benutzerdefinierten Domainnamen oder private DNS-Namen verwendet.	Diese Architektur führt zu einer Diskrepanz zwischen dem Host-Header und dem im TLS-Handshake verwendeten SNI und überwindet die Domain-Front-Einschränkungen CloudFront nicht. Wir empfehlen Ihnen, Ihre VPC zur Verwendung von privatem DNS zu migrieren.

Architektur	Vorgeschlagene Migration
Verwenden Sie Domain-Sharding, um Inhalte auf mehrere Domains oder Subdomains zu verteilen.	Diese Architektur führt zu einer Diskrepanz zwischen dem Host-Header und dem im TLS-Handshake verwendeten SNI und CloudFront überwindet die Domain-Front-Einschränkungen nicht. Wir empfehlen, dieses Anti-Pattern zu verwenden HTTP/2 und es zu entfernen.

Im Folgenden finden Sie Überlegungen zur Verwendung des Endpunktzugriffsmodus:

- Wenn der Endpunktzugriffsmodus einer API oder eines Domainnamens lautet `STRICT`, können Sie den Endpunkttyp nicht ändern. Um den Endpunkttyp zu ändern, ändern Sie zunächst den Endpunktzugriffsmodus auf `BASIC`.
- Nachdem Sie den Endpunktzugriffsmodus von `BASIC` auf geändert haben `STRICT`, gibt es eine Verzögerung von 15 Minuten, bis API Gateway den strikten Endpunktzugriffsmodus erzwingt.
- Wenn Sie eine Sicherheitsrichtlinie von einer Richtlinie, die mit `SecurityPolicy_` beginnt, zu einer älteren Richtlinie ändern, müssen Sie den Endpunktzugriffsmodus auf `""` deaktivieren.

Überlegungen

Im Folgenden finden Sie Überlegungen zu Sicherheitsrichtlinien für REST-APIs in API Gateway:

- Sie können die Sicherheitsrichtlinie in eine OpenAPI-Definitionsdatei importieren. Weitere Informationen finden Sie unter [x-amazon-apigateway-endpoint-Zugriffsmodus](#).
- Ihre API kann einem benutzerdefinierten Domainnamen mit einer anderen Sicherheitsrichtlinie als Ihrer API zugeordnet werden. Wenn Sie diesen benutzerdefinierten Domainnamen aufrufen, verwendet API Gateway die Sicherheitsrichtlinie der API, um den TLS-Handshake auszuhandeln. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.
- Wenn Sie Ihre Sicherheitsrichtlinie ändern, dauert es etwa 15 Minuten, bis das Update abgeschlossen ist. Sie können die `apiStatus` Ihrer API überwachen. Wenn Ihre API aktualisiert wird, ist `apiStatus` `UPDATING` und wenn sie abgeschlossen ist, wird sie es auch sein `AVAILABLE`. Wenn Ihr API-Status lautet `UPDATING`, können Sie ihn trotzdem aufrufen.

- API Gateway unterstützt Sicherheitsrichtlinien für alle APIs. Sie können jedoch nur eine Sicherheitsrichtlinie für REST-APIs auswählen. API Gateway unterstützt nur die TLS_1_2 Sicherheitsrichtlinie für HTTP oder WebSocket APIs.
- Sie können die Sicherheitsrichtlinie für eine API nicht von TLS_1_0 bis aktualisieren TLS_1_2.
- Einige Sicherheitsrichtlinien unterstützen sowohl ECDSA- als auch RSA-Cipher Suites. Wenn Sie diese Art von Richtlinie mit einem benutzerdefinierten Domännennamen verwenden, entsprechen die Cipher Suites dem vom Kunden bereitgestellten Zertifikatsschlüsseltyp, entweder RSA oder ECDSA. Wenn Sie diesen Richtlinientyp mit einer REST-API verwenden, entsprechen die Cipher Suites den Cipher Suites, die mit RSA-Zertifikatstypen kompatibel sind.

Unterstützte Sicherheitsrichtlinien

In den folgenden Tabellen werden die [Sicherheitsrichtlinien](#) beschrieben, die für jeden REST-API-Endpunkttyp und benutzerdefinierten Domainnamentyp angegeben werden können. Mit diesen Richtlinien können Sie eingehende Verbindungen steuern. API Gateway unterstützt nur TLS 1.2 beim Ausgang. Sie können die Sicherheitsrichtlinie für Ihre API oder Ihren benutzerdefinierten Domainnamen jederzeit aktualisieren.

Richtlinien, die FIPS im Titel enthalten sind, sind mit dem Federal Information Processing Standard (FIPS) kompatibel, einem Standard der US-amerikanischen und kanadischen Regierung, der die Sicherheitsanforderungen für kryptografische Module zum Schutz vertraulicher Informationen festlegt. Weitere Informationen finden Sie unter [Federal Information Processing Standard \(FIPS\) 140](#) auf der Seite AWS Cloud Security Compliance.

Alle FIPS-Richtlinien nutzen das AWS-LC FIPS-validierte kryptografische Modul. Weitere Informationen finden Sie auf der Seite [AWS-LC Cryptographic Module](#) auf der Website des NIST Cryptographic Module Validation Program.

Richtlinien, die PQ im Titel enthalten sind, verwenden [Post-Quantum Kryptografie \(PQC\)](#), um hybride Schlüsselaustauschalgorithmen für TLS zu implementieren, um die Vertraulichkeit des Datenverkehrs vor future Bedrohungen durch Quantencomputer zu gewährleisten.

Richtlinien, die PFS im Titel enthalten sind, verwenden [Perfect Forward Secrecy \(PFS\)](#), um sicherzustellen, dass Sitzungsschlüssel nicht kompromittiert werden.

Richtlinien, die beides FIPS und PQ in ihrem Titel enthalten, unterstützen beide Funktionen.

Standard-Sicherheitsrichtlinien

Wenn Sie eine neue REST-API oder eine benutzerdefinierte Domain erstellen, wird der Ressource eine Standardsicherheitsrichtlinie zugewiesen. Die folgende Tabelle zeigt die Standardsicherheitsrichtlinie für diese Ressourcen.

Ressource	Name der Standard-Sicherheitsrichtlinie		
Regionale APIs	TLS_1_0		
Edge-optimized APIs	TLS_1_0		
Private APIs	TLS_1_2		
Regionale Domäne	TLS_1_2		
Edge-optimized Domäne	TLS_1_2		
Private Domain	TLS_1_2		

Unterstützte Sicherheitsrichtlinien für regionale und private APIs und benutzerdefinierte Domainnamen

In der folgenden Tabelle werden die Sicherheitsrichtlinien beschrieben, die für regionale und private APIs und benutzerdefinierte Domainnamen angegeben werden können:

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS13_1_3_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS13_1_3_FIPS_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384
SecurityPolicy_TLS13_1_2_FIPS_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS13_1_2_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS 13_1_2_PQ_2025_09	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS13_1_2_2021_06	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
TLS_1_2	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
TLS_1_0	TLS1.3	TLS1.3
	TLS1.2 TLS1.1 TLS1.0	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA

Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
		• ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Unterstützte Sicherheitsrichtlinien für Edge-optimierte APIs und benutzerdefinierte Domainnamen

In der folgenden Tabelle werden die Sicherheitsrichtlinien beschrieben, die für Edge-optimierte APIs und Edge-optimierte benutzerdefinierte Domainnamen angegeben werden können:

Name der Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS13_2025_EDGE	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS12_PFS_2025_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Name der Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
		TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305

Name der Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
SecurityPolicy_TLS12_2018_E DGE	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Name der Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
TLS_1_0	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256
	TLS1.1	• TLS_AES_256_GCM_SHA384
	TLS1.0	• TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA

Name der Sicherheitsrichtlinie	Unterstützte TLS-Versionen	Unterstützte Chiffren
		• ECDHE-RSA-AES256-SHA • AES256-SHA256 • AES256-SHA

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für die gleichen Verschlüsselungsverfahren. Die folgende Tabelle ordnet den OpenSSL-Namen dem RFC-Namen für jedes Verschlüsselungsverfahren zu. Weitere Informationen finden Sie unter [Verschlüsselungen](#) in der OpenSSL-Dokumentation.

OpenSSL-Verschlüsselungsname	RFC-Verschlüsselungsname
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

OpenSSL-Verschlüsselungsname	RFC-Verschlüsselungsname
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Wie ändere ich eine Sicherheitsrichtlinie

Sie können die Sicherheitsrichtlinie für Ihre API ändern. Wenn Sie über Ihren benutzerdefinierten Domainnamen Traffic an Ihre APIs senden, müssen die API und der benutzerdefinierte Domainname nicht dieselbe Sicherheitsrichtlinie haben. Wenn Sie diesen benutzerdefinierten Domainnamen aufrufen, verwendet API Gateway die Sicherheitsrichtlinie der API, um den TLS-Handshake auszuhandeln. Aus Konsistenzgründen empfehlen wir jedoch, dass Sie dieselbe Sicherheitsrichtlinie für Ihren benutzerdefinierten Domainnamen und Ihre API verwenden.

Wenn Sie Ihre Sicherheitsrichtlinie ändern, dauert es etwa 15 Minuten, bis das Update abgeschlossen ist. Sie können die `apiStatus` Ihrer API überwachen. Wenn Ihre API aktualisiert wird, ist `apiStatus` `UPDATING` und wenn sie abgeschlossen ist, wird sie es auch `AVAILABLE`. Wenn Ihre API aktualisiert wird, können Sie sie trotzdem aufrufen.

AWS-Managementkonsole

Um die Sicherheitsrichtlinie einer API zu ändern

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie API-Einstellungen und anschließend Bearbeiten aus.
4. Wählen Sie für Sicherheitsrichtlinie eine neue Richtlinie aus, die mit `SecurityPolicy_` beginnt.
5. Wählen Sie für den Endpunktzugriffsmodus die Option `Strict` aus.
6. Wählen Sie `Änderungen speichern` aus.

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden. Da Sie den Endpunktzugriffsmodus auf `Strict` geändert haben, dauert es etwa 15 Minuten, bis die Änderungen vollständig übernommen werden.

AWS CLI

Mit dem folgenden Befehl [update-rest-api](#) wird eine API aktualisiert, sodass sie die Sicherheitsrichtlinie verwendet: `SecurityPolicy_TLS13_1_3_2025_09`

```
aws apigateway update-rest-api \
  --rest-api-id abcd1234 \
  --patch-operations '[
    {
      "op": "replace",
      "path": "/securityPolicy",
      "value": "SecurityPolicy_TLS13_1_3_2025_09"
    },
    {
      "op": "replace",
      "path": "/endpointAccessMode",
      "value": "STRICT"
    }
  ]'
```

```
}  
]'
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "id": "abcd1234",  
  "name": "MyAPI",  
  "description": "My API with a new security policy",  
  "createdDate": "2025-02-04T11:47:06-08:00",  
  "apiKeySource": "HEADER",  
  "endpointConfiguration": {  
    "types": [  
      "REGIONAL"  
    ],  
    "ipAddressType": "dualstack"  
  },  
  "tags": {},  
  "disableExecuteApiEndpoint": false,  
  "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",  
  "endpointAccessMode": "STRICT"  
  "rootResourceId": "efg456"  
}
```

Der folgende Befehl [update-rest-api aktualisiert eine API](#), die eine erweiterte Sicherheitsrichtlinie verwendet hat, um die Sicherheitsrichtlinie zu verwenden. TLS_1_0

```
aws apigateway update-rest-api \  
  --rest-api-id abcd1234 \  
  --patch-operations '[  
    {  
      "op": "replace",  
      "path": "/securityPolicy",  
      "value": "TLS_1_0"  
    },  
    {  
      "op": "replace",  
      "path": "/endpointAccessMode",  
      "value": ""  
    }  
  ]'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "id": "abcd1234",
  "name": "MyAPI",
  "description": "My API with a new security policy",
  "createdDate": "2025-02-04T11:47:06-08:00",
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "REGIONAL"
    ],
    "ipAddressType": "dualstack"
  },
  "tags": {},
  "disableExecuteApiEndpoint": false,
  "securityPolicy": "TLS_1_0",
  "rootResourceId": "efg456"
}
```

IP-Adresstypen für REST-APIs in API Gateway

Beim Erstellen einer API geben Sie den Typ der IP-Adressen an, die Ihre API aufrufen dürfen. Sie können IPv4 auswählen, um IPv4-Adressen für den Aufruf Ihrer API zuzulassen, oder Sie können Dualstack auswählen, um sowohl IPv4- als auch IPv6-Adressen zuzulassen. Wir empfehlen, den IP-Adresstyp auf Dualstack zu setzen, um Engpässe im IP-Adressraum zu vermeiden und Ihre Sicherheitsposition zu verbessern. Weitere Informationen zu den Vorteilen eines Dualstack-IP-Adresstyps finden Sie unter [IPv6 in AWS](#).

Um Ihre API ausschließlich auf IPv6-Datenverkehr zu beschränken, können Sie eine Ressourcensicherheitsrichtlinie erstellen und die Quell-IP-Adressen nur auf IPv6-Bereiche einschränken. Sie können den IP-Adresstyp ändern, indem Sie die API-Konfiguration aktualisieren. Diese Änderung wird sofort wirksam, ein erneutes Bereitstellen der API ist nicht erforderlich. Weitere Informationen finden Sie unter [the section called “Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs”](#).

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können Ihre Verwendung von IP-Adresstypen beeinflussen:

- Der Standard-IP-Adresstyp für alle regionalen und Edge-optimierten APIs ist IPv4.
- Private APIs können nur den IP-Adresstyp Dualstack haben.
- Wenn Sie den IP-Adresstyp einer bestehenden API von IPv4 in Dualstack ändern, stellen Sie sicher, dass alle Richtlinien, die den Zugriff auf Ihre APIs steuern, aktualisiert wurden, um IPv6-Aufrufe zu berücksichtigen. Wenn Sie den IP-Adresstyp ändern, wird die Änderung sofort wirksam.
- Wenn Sie den Endpunkttyp einer API von „Regional“ oder „Edge-optimiert“ zu „Privat“ migrieren, ändert API Gateway den IP-Adresstyp in „Dualstack“. Weitere Informationen finden Sie unter [the section called “Ändern eines öffentlichen oder privaten API-Endpunkttyps”](#).
- Wenn Sie den Endpunkttyp einer API von „Privat“ zu „Regional“ migrieren, müssen Sie den IP-Adresstyp auf „Dualstack“ festlegen. Nach Abschluss der Endpunktmigration können Sie den IP-Adresstyp in IPv4 ändern. Weitere Informationen finden Sie unter [the section called “Ändern eines öffentlichen oder privaten API-Endpunkttyps”](#).
- Ihre API kann einem benutzerdefinierten Domainnamen mit einem anderen IP-Adresstyp als Ihre API zugewiesen werden. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.
- Sie können keine externe Definitionsdatei verwenden, um den IP-Adresstyp Ihrer API zu konfigurieren.

Ändern des IP-Adresstyps einer REST-API

Sie können den IP-Adresstyp ändern, indem Sie die API-Konfiguration aktualisieren. Dies ist über die AWS-Managementkonsole, die AWS CLI, CloudFormation oder ein AWS SDK möglich. Wenn Sie den IP-Adresstyp der API ändern, müssen Sie Ihre API nicht erneut bereitstellen, damit die Änderungen wirksam werden. Stellen Sie vor der Änderung des IP-Adresstyps sicher, dass alle Richtlinien, die den Zugriff auf Ihre APIs steuern, für IPv6-Aufrufe aktualisiert wurden.

AWS-Managementkonsole

So ändern Sie den IP-Adresstyp einer REST-API

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie API-Einstellungen und anschließend Bearbeiten aus.
4. Wählen Sie unter „IP-Adresstyp“ entweder IPv4 oder Dualstack aus.

5. Klicken Sie auf Änderungen speichern.

Die Änderung der API-Konfiguration wird sofort wirksam.

AWS CLI

Der folgende [update-rest-api](#)-Befehl aktualisiert eine API so, dass sie den IP-Adresstyp Dualstack verwendet:

```
aws apigateway update-rest-api \  
  --rest-api-id abcd1234 \  
  --patch-operations "op='replace',path='/endpointConfiguration/  
ipAddressType',value='dualstack'"
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "id": "abcd1234",  
  "name": "MyAPI",  
  "description": "My API with a dualstack IP address type",  
  "createdDate": "2025-02-04T11:47:06-08:00",  
  "apiKeySource": "HEADER",  
  "endpointConfiguration": {  
    "types": [  
      "REGIONAL"  
    ],  
    "ipAddressType": "dualstack"  
  },  
  "tags": {},  
  "disableExecuteApiEndpoint": false,  
  "rootResourceId": "efg456"  
}
```

Methoden für REST APIs in API Gateway

In API Gateway setzt sich eine API-Methode aus einer [Methodenanfrage](#) und eine [Methodenantwort](#) zusammen. Sie richten eine API-Methode ein, um zu definieren, was ein Client tun sollte oder tun muss, um eine Anforderung für den Zugriff auf den Service am Backend zu übermitteln und um die Antworten zu definieren, die der Client im Gegenzug erhält. Für die Eingabe können Sie Methodenanforderungs-Parameter oder eine anwendbare Nutzlast für den Client auswählen, um die

erforderlichen oder optionalen Daten zur Laufzeit bereitzustellen. Für die Ausgabe bestimmen Sie den Statuscode der Methodenantwort, die Header und den geltenden Textkörper als Ziele denen Backend-Antwortdaten zugewiesen werden, bevor die Rückgabe an den Client erfolgt. Um den Client-Developer dabei zu unterstützen, das Verhalten und die Eingabe- und Ausgabeformate Ihrer API zu verstehen, können Sie [Ihre API dokumentieren](#) und [ordnungsgemäße Fehlermeldungen](#) für [ungültige Anforderungen](#) bereitstellen.

Eine API-Methodenanforderung ist eine HTTP-Anforderung. Um die Methodenanforderung einzurichten, konfigurieren Sie eine HTTP-Methode (oder ein Verb), den Pfad zu einer API-[Ressource](#), Header, geltende Parameter für Abfragezeichenfolgen. Außerdem konfigurieren Sie eine Nutzlast, wenn die HTTP-Methode POST, PUT oder PATCH ist. Um beispielsweise mithilfe der [PetStore Beispiel-API](#) ein Haustier abzurufen, definieren Sie die API-Methodenanfrage von GET /pets/{petId}. Dabei {petId} handelt es sich um einen Pfadparameter, der zur Laufzeit eine Zahl annehmen kann.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Wenn der Client einen falschen Pfad angibt, zum Beispiel /pet/1 oder /pets/one anstatt /pets/1, wird eine Ausnahme ausgelöst.

Eine API-Methodenantwort ist eine HTTP-Antwort mit einem bestimmten Statuscode. Für eine Nicht-Proxy-Integration müssen Sie Methodenantworten einrichten, um die erforderlichen oder optionalen Ziele von Mappings anzugeben. Diese transformieren Integrationsantwort-Header oder einen Textkörper in verknüpfte Methodenantwort-Header oder einen Textkörper. Das Mapping kann so einfach wie eine [identische Transformation](#) sein, mit der die Header oder der Text unverändert mittels der Integration übergeben werden. Zum Beispiel zeigt die folgende 200-Methodenantwort ein Beispiel für die Weiterleitung einer unveränderten erfolgreichen Integrationsantwort.

```
200 OK
Content-Type: application/json
...

{
  "id": "1",
  "type": "dog",
  "price": "$249.99"
}
```

In der Regel können Sie eine Methodenantwort entsprechend einer bestimmten Antwort vom Backend definieren. In der Regel handelt es sich dabei um alle 2XX-, 4XX- und 5XX-Antworten. Dies kann jedoch nicht praktikabel sein, da Sie häufig nicht alle Antworten, die ein Backend möglicherweise zurückgibt, im Vorfeld wissen können. In der Praxis können Sie eine Methodenantwort als Standard festlegen, um die unbekannten oder nicht zugewiesenen Antworten vom Backend zu verarbeiten. Es ist ein bewährtes Verfahren, die 500-Antwort als Standard festzulegen. In jedem Fall müssen Sie mindestens eine Methodenantwort für Nicht-Proxy-Integrationen einrichten. Andernfalls gibt API Gateway eine 500-Fehler-Antwort an den Client zurück, selbst wenn die Anfrage am Backend erfolgreich ist.

Zur Unterstützung eines stark typisierten SDKs, z. B. ein Java-SDK für Ihre API, sollten Sie das Datenmodell zur Eingabe für die Methodenanforderung und das Datenmodell zur Ausgabe der Methodenantwort definieren.

Voraussetzungen

Vor dem Einrichten einer API-Methode, überprüfen Sie Folgendes:

- Die Methode muss in API Gateway verfügbar sein. Folgen Sie den Anweisungen in [Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen](#).
- Wenn die Methode mit einer Lambda-Funktion kommunizieren soll, müssen Sie die Lambda-Aufrufrolle und die Lambda-Ausführungsrolle bereits in IAM erstellt haben. Sie müssen außerdem die Lambda-Funktion erstellt haben, mit der Ihre Methode in AWS Lambda kommuniziert. Befolgen Sie zum Erstellen der Rollen und der Funktion die Anweisungen in [Lambda-Funktion für die Nicht-Proxy-Integration von Lambda erstellen](#) unter [Wählen Sie ein AWS Lambda Integrations-Tutorial](#).
- Wenn Sie möchten, dass die Methode mit einer HTTP- oder HTTP-Proxy-Integration kommunizieren soll, muss die HTTP-Endpunkt-URL, mit der die Methode kommuniziert, bereits erstellt sein und Sie müssen Zugriff darauf haben.
- Stellen Sie sicher, dass Ihre Zertifikate für HTTP- und HTTP-Proxy-Endpunkte von API Gateway unterstützt werden. Weitere Informationen finden Sie unter [Von API Gateway unterstützte Zertifizierungsstellen für HTTP- und HTTP-Proxy-Integrationen](#).

Themen

- [Methodenanfrage in API Gateway einrichten](#)
- [Einrichten einer Methodenantwort in API Gateway](#)
- [Methode über die API Gateway-Konsole einrichten](#)

Methodenanfrage in API Gateway einrichten

Zum Einrichten einer Methodenanforderung müssen nach dem Erstellen einer [RestApi](#)Ressource die folgenden Aufgaben ausgeführt werden:

1. Erstellen einer neuen API oder Auswahl einer vorhandenen API-[Ressourcen](#)-Entität.
2. Erstellen einer API-[Methoden](#)-Ressource, die ein spezifisches HTTP-Verb für die neue oder ausgewählte API-Ressource ist. Diese Aufgabe kann noch weiter in die folgenden Sub-Aufgaben unterteilt werden:
 - Hinzufügen einer HTTP-Methode zur Methodenanforderung
 - Konfigurieren von Anforderungsparametern
 - Definieren eines Modells für den Anforderungstextkörper
 - Einfügen eines Autorisierungsschemas
 - Aktivieren einer Anforderungsvalidierung

Sie können diese Aufgaben mithilfe der folgenden Methoden durchführen:

- [API Gateway-Konsole](#)
- AWS CLI [Befehle \(create-resource und put-method\)](#)
- AWS [SDK-Funktionen](#) (z. B. in Node.js, `createResource` und `PutMethod`)
- API Gateway-REST-API ([resource:create](#) und [method:put](#)).

Themen

- [Einrichten von API-Ressourcen](#)
- [Einrichten einer HTTP-Methode](#)
- [Einrichten von Methodenanforderungs-Parametern](#)
- [Einrichten eines Methodenanforderungsmodells](#)
- [Einrichten der Autorisierung der Methodenanforderung](#)
- [Einrichten einer Validierung der Methodenanforderung](#)

Einrichten von API-Ressourcen

In einer API Gateway-API setzen Sie adressierbare Ressourcen als eine API-[Ressourcen](#)-Entitätsstruktur ein, mit der Stammressource (/) oben in der Hierarchie. Die Stammressource bezieht

sich auf die Basis-URL der API, die aus dem API-Endpunkt und einem Stufennamen besteht. In der API Gateway-Konsole wird diese Basis-URI als Invoke URI (Aufruf-URI) bezeichnet und im Stage-Editor der API angezeigt, nachdem die API bereitgestellt wurde.

Der API-Endpunkt kann ein Standard-Host-Name oder ein benutzerdefinierter Domänenname sein. Der Standard-Host-Name hat das folgende Format:

```
{api-id}.execute-api.{region}.amazonaws.com
```

In diesem Format `{api-id}` steht der für die API-ID, die von API Gateway generiert wird. Die `{region}`-Variable stellt die AWS -Region (z. B. `us-east-1`) dar, die Sie bei der Erstellung der API auswählen. Ein benutzerdefinierter Domänenname ist jeder benutzerfreundliche Name in einer gültigen Internetdomäne. Wenn Sie beispielsweise eine Internetdomäne von `example.com` registriert haben, ist alles von `*.example.com` ein gültiger benutzerdefinierter Domänenname. Weitere Informationen finden Sie unter [Erstellen eines benutzerdefinierten Domänenname](#).

Bei der [PetStore Beispiel-API](#) macht die Root-Ressource (`/`) die Tierhandlung verfügbar. Die `/pets`-Ressource stellt die Sammlung von Haustieren, die im PetStore verfügbar sind, dar. Die `/pets/{petId}` stellt ein einzelnes Haustier einer bestimmten ID (`petId`) dar. Der Pfadparameter von `{petId}` ist Teil der Anforderungsparameter.

Zum Konfigurieren einer API-Ressource wählen Sie eine vorhandene Ressource als übergeordnetes Element aus und erstellen anschließend unter dieser die untergeordnete Ressource. Sie beginnen mit der Stammressource als übergeordnetes Element, fügen diesem eine Ressource hinzu, fügen eine andere Ressource dieser untergeordneten Ressource als neues übergeordnetes Element und so weiter der übergeordneten ID hinzu. Anschließend fügen Sie die benannte Ressource dem übergeordneten Element hinzu.

Mit dem folgenden Befehl [get-resources](#) werden alle Ressourcen einer API abgerufen:

```
aws apigateway get-resources --rest-api-id apiId
```

Für die PetStore Beispiel-API sieht die Ausgabe wie folgt aus:

```
{
  "items": [
    {
      "path": "/pets",
      "resourceMethods": {
        "GET": {}
      }
    }
  ]
}
```

```

    },
    "id": "6sxx2j",
    "pathPart": "pets",
    "parentId": "svzr2028x8"
  },
  {
    "path": "/pets/{petId}",
    "resourceMethods": {
      "GET": {}
    },
    "id": "rjkmth",
    "pathPart": "{petId}",
    "parentId": "6sxx2j"
  },
  {
    "path": "/",
    "id": "svzr2028x8"
  }
]
}

```

Jedes Element listet die ID der Ressource (`id`) und, mit Ausnahme der Stammressource, das unmittelbar übergeordnete Element (`parentId`) sowie den Ressourcennamen (`pathPart`). Die Stammressource ist dahingehend besonders, dass sie über kein übergeordnetes Element verfügt. Nachdem Sie eine Ressource als übergeordnetes Element ausgewählt haben, verwenden Sie den folgenden Befehl, um eine untergeordnete Ressource hinzuzufügen.

```

aws apigateway create-resource --rest-api-id apiId \
  --parent-id parentId \
  --path-part resourceName

```

Um beispielsweise Tiernahrung zum Verkauf auf der PetStore Website hinzuzufügen, verwenden Sie den folgenden Befehl:

```

aws apigateway create-resource --rest-api-id a1b2c3 \
  --parent-id svzr2028x8 \
  --path-part food

```

Die Ausgabe sieht wie folgt aus:

```
{
```

```
"path": "/food",
"pathPart": "food",
"id": "xdsvhp",
"parentId": "svzr2028x8"
}
```

Verwenden Sie eine Proxy-Ressource zur Optimierung der API-Einrichtung

Wenn das Geschäft wächst, kann der PetStore Eigentümer beschließen, Lebensmittel, Spielzeug und andere Artikel für Haustiere zum Verkauf anzubieten. Um dies zu unterstützen, können Sie /food, /toys und andere Ressourcen unter der Stammressource hinzufügen. Unter jeder Verkaufskategorie möchten Sie möglicherweise auch weitere Ressourcen hinzufügen, wie /food/{type}/{item}, /toys/{type}/{item} usw. Dies kann sehr zeitaufwändig werden. Wenn Sie sich dazu entscheiden, eine mittlere Schicht {subtype} den Ressourcenpfaden hinzuzufügen, um die Pfadhierarchie in /food/{type}/{subtype}/{item}, /toys/{type}/{subtype}/{item} usw. zu ändern, könnten die Änderungen zu Fehlern bei der vorhandenen API-Einrichtung führen. Um dies zu vermeiden, können Sie ein API Gateway-[Proxy-Ressource](#) verwenden, um einen Satz von API-Ressourcen auf einmal bereitzustellen.

API Gateway definiert eine Proxy-Ressource als Platzhalter für eine Ressource, die bei der Übertragung der Anfrage angegeben wird. Eine Proxy-Ressource wird durch einen speziellen Pfadparameter {proxy+} ausgedrückt, häufig auch als gieriger Pfadparameter bezeichnet. Das +-Zeichen gibt an, welche untergeordneten Ressourcen ihm angehängt werden. Der Platzhalter /parent/{proxy+} steht für jede Ressource, die mit dem Pfadmuster /parent/* übereinstimmt. Sie können eine beliebige Zeichenfolge für den Namen des Greedy-Pfadparameter verwenden.

Der folgende [create-resource](#)-Befehl erstellt eine Proxy-Ressource unter dem Stammverzeichnis (/ {proxy+}):

```
aws apigateway create-resource --rest-api-id apiId \
  --parent-id rootResourceId \
  --path-part {proxy+}
```

Die Ausgabe sieht wie folgt aus:

```
{
  "path":("/{proxy+}",
  "pathPart": "{proxy+}",
  "id": "234jdr",
```

```
"parentId": "svzr2028x8"
}
```

Für das PetStore-API-Beispiel können Sie `/proxy+` verwenden, um die `/pets` und `/pets/{petId}` darzustellen. Diese Proxyressource kann auch auf alle anderen (vorhandenen oder to-be-added) Ressourcen verweisen `/food/{type}/{item}/toys/{type}/{item}`, z. B. auf, usw. oder `/food/{type}/{subtype}/{item}/toys/{type}/{subtype}/{item}`, usw. Die Backend-Developer bestimmt die Ressourcenhierarchie und der Client-Developer ist dafür verantwortlich, sie zu verstehen. API Gateway übergibt einfach alles, was der Client an das Backend übermittelt hat.

Eine API kann über mehr als eine Proxy-Ressource verfügen. Beispielsweise sind die folgenden Proxy-Ressourcen innerhalb einer API erlaubt, vorausgesetzt `/parent/{proxy+}` ist nicht dieselbe übergeordnete Ressource wie `/parent/{child}/{proxy+}`.

```
/proxy+
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Wenn eine Proxy-Ressource über gleichwertige Nicht-Proxy-Elemente verfügt, werden die gleichwertigen Ressourcen von der Darstellung der Proxy-Ressource ausgeschlossen. Bei den vorherigen Beispielen bezieht sich `/proxy+` auf alle Ressourcen unter der Stammressource mit Ausnahme der `/parent[/*]`-Ressourcen. Mit anderen Worten, eine Methodenanforderung für eine bestimmte Ressource hat Vorrang vor einer ähnlichen Methodenanforderung für eine generische Ressource auf derselben Ebene der Ressourcenhierarchie.

Die folgende Tabelle zeigt, wie API Gateway Anfragen für die prod-Stufe einer API an die folgenden Ressourcen weiterleitet.

```
ANY /proxy+
GET /pets/{proxy+}
GET /pets/dog
```

Anforderung	Ausgewählte Route	Erklärung
GET https:// <i>api-id</i> .execute-	GET /pets/dog	Die Anforderung stimmt vollständig mit dieser Ressource überein.

Anforderung	Ausgewählte Route	Erklärung
api. <i>region</i> .amazonaws.com/prod/pets/dog		
GET https://api- <i>id</i> .execute-api. <i>region</i> .amazonaws.com/prod/pets/cats	GET /pets/{proxy+}	Die Greedy-Pfadvariable /pets/{proxy+} fängt diese Anfrage ab.
GET https://api- <i>id</i> .execute-api. <i>region</i> .amazonaws.com/prod/animals	GET /{proxy+}	Die Greedy-Pfadvariable /{proxy+} fängt diese Anfrage ab.

Eine Proxy-Ressource kann über keine untergeordnete Ressource verfügen. Jede API-Ressource nach {proxy+} ist überflüssig und zweideutig. Die folgenden Proxy-Ressourcen sind nicht innerhalb einer API erlaubt.

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Einrichten einer HTTP-Methode

Eine API-Methodenanforderung wird durch die API Gateway-[Methode](#)-Ressource gekapselt. Um die Methodenanforderung einzurichten, müssen Sie zuerst die Method-Ressource instanziiieren und mindestens eine HTTP-Methode und einen Autorisierungstyp für die Methode festlegen.

API Gateway ist eng mit der Proxy-Ressource verbunden und unterstützt die HTTP-API-Methode ANY. Diese ANY-Methode entspricht der HTTP-Methode, die zur Laufzeit bereitgestellt wird. Sie gibt Ihnen die Möglichkeit, eine einzige API-Methodeneinrichtung für die unterstützten HTTP-Methoden DELETE, GET, HEAD, OPTIONS, PATCH, POST und PUT zu verwenden.

Sie können die ANY-Methode auch auf einer Nicht-Proxy-Ressource einrichten. Durch die Kombination der ANY-Methode mit einer Proxy-Ressource erhalten Sie eine einzige API-Methodeneinrichtung für alle unterstützten HTTP-Methoden gegen alle Ressourcen einer API.

Darüber hinaus kann sich das Backend weiterentwickeln, ohne die vorhandene API-Einrichtung zu stören.

Berücksichtigen Sie vor dem Einrichten einer API-Methode, wer die Methode aufrufen kann. Legen Sie den Autorisierungstyp entsprechend Ihres Plans fest. Für einen offenen Zugriff setzen Sie ihn auf NONE. Zur Nutzung von IAM-Berechtigungen setzen Sie den Autorisierungstyp auf AWS_IAM. Um eine Lambda-Genehmiger-Funktion zu verwenden, setzen Sie diese Eigenschaft auf CUSTOM. Um einen Amazon Cognito-Benutzerpool zu verwenden, setzen Sie den Autorisierungstyp auf COGNITO_USER_POOLS.

Der folgende [put-method](#)-Befehl erstellt eine Methodenanforderung für das ANY-Verb und verwendet IAM-Berechtigungen zur Zugriffskontrolle.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method ANY \  
  --authorization-type AWS_IAM
```

Informationen zur Erstellung einer API-Methodenanforderung mit einem anderen Autorisierungstyp finden Sie unter [the section called “Einrichten der Autorisierung der Methodenanforderung”](#).

Einrichten von Methodenanforderungs-Parametern

Methodenanforderungs-Parameter sind eine Möglichkeit für einen Client, Eingabedaten oder Ausführungskontext bereitzustellen, damit die Methodenanforderung abgeschlossen werden kann. Ein Methodenparameter kann ein Pfadparameter, ein Header oder ein Abfragezeichenfolge-Parameter sein. Im Rahmen der Methodenanforderungseinrichtung müssen Sie die erforderlichen Anforderungsparameter deklarieren, um sie dem Client zur Verfügung zu stellen. Für eine Nicht-Proxy-Integration können Sie diese Anforderungsparameter in ein Formular umwandeln, das mit der Backend-Anforderung kompatibel ist.

Zum Beispiel ist für die GET /pets/{petId}-Methodenanforderung die {petId}-Pfadvariable ein erforderlicher Anforderungsparameter. Sie können diesen Pfadparameter beim Aufrufen des Befehls AWS CLI der put-method deklarieren. Der folgende [put-method](#)-Befehl erstellt eine Methode mit einem erforderlichen Pfadparameter:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id rjkmth \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --path-parameters petId
```

```
--request-parameters method.request.path.petId=true
```

Wenn ein Parameter nicht erforderlich ist, können Sie ihn auf `false` in `request-parameters` setzen. Wenn beispielsweise die `GET /pets`-Methode einen optionalen Abfragezeichenfolgenparameter `type` und einen optionalen Header-Parameter `age` verwendet, können Sie diese mit dem folgenden [put-method](#)-Befehl deklarieren:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxz2j \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters  
  method.request.querystring.type=false,method.request.header.age=false
```

Anstelle dieser Kurzform können Sie eine JSON-Zeichenfolge verwenden, um den `request-parameters`-Wert zu setzen:

```
'{"method.request.querystring.type":false,"method.request.header.age":false}'
```

Mit dieser Einrichtung kann der Client die Haustiere nach Typ abfragen:

```
GET /pets?type=dog
```

Und der Client kann Hunde, die Welpen sind, wie folgt abfragen:

```
GET /pets?type=dog  
age:puppy
```

Weitere Informationen, wie Methodenanforderungs-Parameter den Integrationsanforderungs-Parametern zugewiesen werden, finden Sie unter [the section called “Integrationen”](#).

Einrichten eines Methodenanforderungsmodells

Für eine API-Methode, die Eingabedaten in eine Nutzlast übernehmen kann, können Sie ein Modell verwenden. Ein Modell wird in einem [JSON-Schema Entwurf 4](#) ausgedrückt und beschreibt die Datenstruktur des Anforderungstexts. Mit einem Modell kann ein Client festlegen, wie eine Methodenanforderungsnutzlast als Eingabe erstellt wird. Noch wichtiger ist, dass API Gateway das Modell verwendet, um [eine Anfrage](#) zu validieren, [ein SDK zu generieren](#) und eine Mapping-Vorlage

für die Einrichtung der Integration in der API Gateway-Konsole zu initialisieren. Informationen zum Erstellen eines [Modells](#) finden Sie unter [Grundlegendes zu Datenmodellen](#).

Je nach Inhaltstypen, kann eine Methodennutzlast über unterschiedliche Formate verfügen. Ein Modell wird für den Medientyp der angewendeten Nutzlast indiziert. API Gateway verwendet den Content-Type Anforderungsheader, um den Inhaltstyp zu bestimmen. Um Methodenanforderungsmodelle einzurichten, fügen Sie der `requestModels` Map beim Aufrufen des Befehls Schlüssel-Wert-Paare des `"media-type": "model-name"` Formats hinzu. AWS CLI `put-method`

Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

[Um beispielsweise ein Modell für die JSON-Nutzlast der POST /pets Methodenanforderung der PetStore Beispiel-API festzulegen, können Sie den folgenden Befehl `put-method` verwenden:](#)

```
aws apigateway put-method \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method POST \  
  --authorization-type "NONE" \  
  --request-models '{"application/json":"petModel"}'
```

Hier ist `petModel` der `name`-Eigenschaftswert einer [Model](#)-Ressource, die ein Haustier beschreibt. Die tatsächliche Schemadefinition wird als JSON-Zeichenfolgenwert der [schema](#)-Eigenschaft der `Model`-Ressource ausgedrückt.

In einem Java oder anderen stark typisierten SDK der API werden die Eingabedaten in die `petModel`-Klasse umgewandelt, die von der Schemadefinition abgeleitet sind. Mit dem Anforderungsmodell werden die Eingabedaten in dem generierten SDK in die `Empty`-Klasse umgewandelt, die von dem Standard-`Empty`-Modell abgeleitet wird. In diesem Fall kann der Client nicht die richtige Datenklasse instanziiieren, um die benötigte Ausgabe bereitzustellen.

Einrichten der Autorisierung der Methodenanforderung

Um zu steuern, wer die API-Methode aufrufen kann, können Sie den [Autorisierungstyp](#) für die Methode konfigurieren. Sie können diesen Typ verwenden, um einen der unterstützten Genehmiger einzusetzen, einschließlich IAM-Rollen und -Richtlinien (`AWS_IAM`), einen Amazon Cognito-Benutzerpool (`COGNITO_USER_POOLS`) oder einen Lambda-Genehmiger (`CUSTOM`).

Um IAM-Berechtigungen für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf **AWS_IAM**. Wenn Sie diese Option festlegen, verifiziert API Gateway die Signatur des Aufrufers in der Anfrage basierend auf den Anmeldeinformationen des Aufrufers. Wenn der bestätigte Benutzer über die Berechtigung zum Aufrufen der Methode verfügt, akzeptiert sie die Anforderung. Andernfalls lehnt sie die Anforderung ab und der Aufrufer erhält die Fehlerantwort „Unbefugt“. Der Aufruf der Methode ist nicht erfolgreich, es sei denn, der Aufrufer hat die Berechtigung, die API-Methode aufzurufen. Die folgende IAM-Richtlinie gewährt dem Aufrufer die Berechtigung, alle darin API-Methoden aufzurufen, die im gleichen AWS-Konto erstellt wurden:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": "arn:aws:execute-api:*:*:*"
    }
  ]
}
```

Weitere Informationen finden Sie unter [the section called “Verwenden von IAM-Berechtigungen”](#).

Derzeit können Sie diese Richtlinie nur Benutzern, Gruppen und Rollen innerhalb des AWS-Konto des API-Besitzers gewähren. Benutzer aus anderen Ländern AWS-Konto können die API-Methoden nur aufrufen, wenn sie eine Rolle innerhalb des API-Besitzers AWS-Konto übernehmen dürfen und über die erforderlichen Berechtigungen verfügen, um die Aktion aufzurufen. `execute-api:Invoke` Weitere Informationen zu kontoübergreifenden Berechtigungen finden Sie unter [Verwenden von IAM-Rollen](#).

Sie können ein AWS SDK oder einen REST-API-Client wie [Postman](#) verwenden AWS CLI, der [Signature Version 4 \(Sigv4\) -Signaturen](#) implementiert.

Um einen Lambda-Genehmiger für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf **CUSTOM** und die [authorizer-id](#)-

Eingabeeigenschaft auf den [id](#)-Eigenschaftswert eines Lambda-Genehmigers, der bereits vorhanden ist. Der referenzierte Lambda-Genehmiger kann vom Typ TOKEN oder REQUEST sein. Informationen zur Erstellung eines Lambda-Genehmigers finden Sie unter [the section called “Lambda-Genehmiger verwenden”](#).

Um einen Amazon Cognito-Benutzerpool für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf `COGNITO_USER_POOLS` und die [authorizer-id](#)-Eingabeeigenschaft auf den [id](#)-Eigenschaftswert des `COGNITO_USER_POOLS`-Genehmigers, der bereits vorhanden ist. Informationen über die Erstellung eines Genehmigers für einen Amazon Cognito-Benutzerpool finden Sie unter [the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”](#).

Einrichten einer Validierung der Methodenanforderung

Sie können die Anforderungsvalidierung beim Einrichten einer API-Methodenanforderung aktivieren. Sie müssen zuerst einen [Anfragevalidierer](#) erstellen: Der folgende [create-request-validator](#)-Befehl erstellt einen Request-Validator, der nur für den Textkörper bestimmt ist.

```
aws apigateway create-request-validator \  
  --rest-api-id 7zw9uyk9kl \  
  --name bodyOnlyValidator \  
  --validate-request-body \  
  --no-validate-request-parameters
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "validateRequestParameters": false,  
  "validateRequestBody": true,  
  "id": "jgppy6",  
  "name": "bodyOnlyValidator"  
}
```

Sie können diesen Anfragevalidierer verwenden, um die Anforderungsvalidierung als Teil der Einrichtung der Methodenanforderung zu nutzen. Der folgende [put-method](#)-Befehl erstellt eine Methodenanforderung, bei der der eingehende Anfrage-Body dem `PetModel` entsprechen muss und zwei Anfrageparameter enthält, die nicht erforderlich sind:

```
aws apigateway put-method \  
  --rest-api-id 7zw9uyk9kl \  
  --method-name GET
```

```
--resource-id xdsvhv \
--http-method PUT \
--authorization-type "NONE" \
--request-parameters '{"method.request.querystring.type": false,
"method.request.querystring.page":false}' \
--request-models '{"application/json":"petModel"}' \
--request-validator-id jgppy6
```

Um einen Anfrageparameter in die Anfragevalidierung einzubeziehen, müssen Sie `validateRequestParameters` für den Anfragevalidierer auf `true` setzen und den jeweiligen Anfrageparameter im `put-method`-Befehl ebenfalls auf `true` setzen.

Einrichten einer Methodenantwort in API Gateway

Eine API-Methodenantwort kapselt die Ausgabe einer API-Methodenanforderung, die der Client erhält. Die Ausgabedaten enthalten einen HTTP-Statuscode, einige Header und in der Regel einen Textkörper.

Mit Nicht-Proxy-Integrationen können die angegebenen Antwortparameter und Textkörper von den zugeordneten Integrationsantwortdaten oder bestimmten statischen Werten entsprechend der Mappings zugewiesen werden. Diese Mappings werden in der Integrationsantwort beschrieben. Das Mapping kann eine identische Transformation sein, die die Integrationsantwort unverändert weiterleitet.

Bei einer Proxy-Integration leitet API Gateway die Antwort des Backends automatisch an die Antwort der Methode weiter. Sie müssen sich nicht um die Einrichtung der API-Methodenantwort kümmern. Bei der Lambda-Proxy-Integration muss die Lambda-Funktion jedoch ein Ergebnis in [diesem Ausgabeformat](#) zurückgeben, damit API Gateway die Integrationsantwort erfolgreich einer Methodenantwort zuordnen kann.

[Programmatisch gesehen läuft die Einrichtung der Methodenantwort darauf hinaus, eine `MethodResponseResource` von API Gateway zu erstellen und die Eigenschaften von `statusCode`, `ResponseParameters` und `ResponseModels` festzulegen.](#)

Wenn Sie Statuscodes für eine API-Methode festlegen, sollten Sie einen als Standard auswählen, um jede Integrationsantwort eines unerwarteten Statuscodes zu bearbeiten. Es ist sinnvoll, 500 als Standard festzulegen, da dies mit dem Umwandeln von ansonsten nicht zugeordneten Antworten als serverseitiger Fehler gleichzusetzen ist. Zur Vereinfachung legt die API Gateway-Konsole die 200-Antwort als Standard fest. Sie können dies jedoch auf die 500-Antwort zurücksetzen.

Um eine Methodenantwort einzurichten, müssen Sie die Methodenanforderung erstellt haben.

Einrichten des Statuscodes der Methodenantwort

Der Statuscode einer Methodenantwort definiert einen Antworttyp. Beispiel: Antworten von 200, 400 und 500 geben jeweils erfolgreiche, clientseitige Fehler und serverseitige Fehlerantworten an.

Um den Statuscode einer Methodenantwort einzurichten, legen Sie die [statusCode](#)-Eigenschaft auf einen HTTP-Statuscode fest. [put-method-response](#)Der folgende Befehl erstellt eine 200 Methodenantwort.

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --status-code 200
```

Einrichten von Parametern der Methodenantwort

Parameter der Methodenantwort definieren, welche Header der Client als Reaktion auf die zugeordnete Methodenanforderung erhält. Antwortparameter legen auch ein Ziel fest, dem API Gateway einen Parameter der Integrationsantwort entsprechend der Mappings zuweist, die in der Integrationsantwort der API beschrieben sind.

Um die Parameter der Methodenantwort einzurichten, fügen Sie der [responseParameters](#)-Zuordnung der MethodResponse-Schlüssel-Wert-Paare im Format "{parameter-name}": "{boolean}" hinzu. Der folgende [put-method-response](#)Befehl legt den my-header Header fest.

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --status-code 200 \  
  --response-parameters method.response.header.my-header=false
```

Einrichten von Modellen der Methodenantwort

Ein Modell der Methodenantwort definiert ein Format des Textkörpers der Methodenantwort. Das Einrichten eines Methodenantwort-Modells ist erforderlich, wenn Sie einen stark typisierten SDK für die API generieren. Es stellt sicher, dass die Ausgabe an eine geeignete Klasse in Java oder Objective-C übergeben wird. In anderen Fällen ist das Einrichten eines Modells optional.

Bevor Sie das Antwortmodell einrichten können, müssen Sie das Modell zunächst in API Gateway erstellen. Dazu können Sie den Befehl [create-model](#) aufrufen. Der folgende [create-model](#)-Befehl erstellt das PetStorePet-Modell, um den Body der Antwort auf die Methodenanforderung GET / pets/{petId} zu beschreiben.

```
aws apigateway create-model \  
  --rest-api-id vaz7da96z6 \  
  --content-type application/json \  
  --name PetStorePet \  
  --schema '{ \  
    "$schema": "http://json-schema.org/draft-04/schema#", \  
    "title": "PetStorePet", \  
    "type": "object", \  
    "properties": { \  
      "id": { "type": "number" }, \  
      "type": { "type": "string" }, \  
      "price": { "type": "number" } \  
    } \  
  }'
```

Das Ergebnis wird als eine API Gateway-[Model](#)-Ressource erstellt.

Um die Methodenantwortmodelle zur Definition des Payload-Formats einzurichten, fügen Sie der Ressourcenzuweisung das Schlüssel-Wert-Paar „application/json“:“ PetStorePet ”hinzu. [requestModelsMethodResponse](#) Der folgende [put-method-response](#) Befehl erstellt eine Methodenantwort, die ein Antwortmodell verwendet, um das Payload-Format zu definieren:

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --status-code 200 \  
  --response-parameters method.response.header.my-header=false \  
  --response-models '{"application/json":"PetStorePet"}'
```

Methode über die API Gateway-Konsole einrichten

Wenn Sie eine Methode mithilfe der REST-API-Konsole erstellen, konfigurieren Sie sowohl die Integrationsanforderung als auch die Methodenanforderung. API Gateway erstellt standardmäßig die 200-Methodenantwort für Ihre Methode.

In den folgenden Anleitungen sehen Sie, wie Sie die Einstellungen für Methodenanfragen bearbeiten und zusätzliche Methodenantworten für Ihre Methode erstellen.

Themen

- [Bearbeiten einer API-Gateway-Methodenanforderung in der API-Gateway-Konsole](#)
- [API Gateway-Methodenantwort über die API Gateway-Konsole einrichten](#)

Bearbeiten einer API-Gateway-Methodenanforderung in der API-Gateway-Konsole

In diesen Anleitungen wird vorausgesetzt, dass Sie Ihre Methodenanforderung bereits erstellt haben. Weitere Informationen zum Erstellen einer Methode finden Sie unter [the section called “Einrichten einer API-Integrationsanforderung mit der Konsole”](#).

1. Wählen Sie Ihre Methode im Bereich Ressourcen aus und klicken Sie dann auf die Registerkarte Methodenanforderung.
2. Wählen Sie im Abschnitt Method request settings (Einstellungen der Methodenanforderung) die Option Edit (Bearbeiten) aus.
3. Wählen Sie für Authorization (Autorisierung) einen verfügbaren Genehmiger aus.
 - a. Um einen offenen Zugriff auf die Methode für alle Benutzer zu aktivieren, wählen Sie None (Keine) aus. Dieser Schritt kann übersprungen werden, wenn die Standardeinstellung nicht geändert wurde.
 - b. Um IAM-Berechtigungen für die Steuerung des Client-Zugriffs auf die Methode zu verwenden, wählen Sie AWS_IAM aus. Mit dieser Auswahl können nur Benutzer der IAM-Rollen mit der korrekten angefügten IAM-Richtlinie diese Methode aufrufen.

Legen Sie zum Erstellen der IAM-Rolle eine Zugriffsrichtlinie mit einem Format wie folgt fest:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ]
    }
  ]
}
```

```
    "Resource": [
      "arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/"
    ]
  }
}
```

In dieser Zugriffsrichtlinie ist `arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/` der ARN Ihrer Methode. Sie finden den ARN Ihrer Methode, indem Sie die Methode auf der Seite Ressourcen auswählen. Weitere Informationen zum Festlegen der IAM-Berechtigungen finden Sie unter [Zugriffskontrolle für eine API mit IAM-Berechtigungen](#).

Sie können die Anleitungen im folgenden Tutorial (???) anpassen, um die IAM-Rolle zu erstellen.

- c. Um einen Lambda-Genehmiger zu verwenden, wählen Sie ein Token oder einen Genehmiger der Anforderung aus. Erstellen Sie einen Lambda-Genehmiger, damit diese Auswahl im Dropdown-Menü angezeigt wird. Informationen über die Erstellung eines Lambda-Genehmigers finden Sie unter [API Gateway-Lambda-Genehmiger verwenden](#).
- d. Um einen Amazon Cognito-Benutzerpool zu verwenden, wählen Sie einen verfügbaren Benutzerpool unter Cognito user pool authorizers (Cognito-Benutzerpool-Genehmiger) aus. Erstellen Sie einen Benutzerpool in Amazon Cognito und einen Amazon Cognito-Benutzerpool-Genehmiger in API Gateway, damit diese Auswahl im Dropdown-Menü angezeigt wird. Informationen darüber, wie Sie einen Amazon Cognito-Benutzerpool-Genehmiger erstellen, finden Sie unter [Steuern Sie den Zugriff auf REST APIs mithilfe von Amazon Cognito Cognito-Benutzerpools als Autorisierer](#).
4. Um die Anforderungsvalidierung anzugeben, wählen Sie einen Wert aus dem Dropdown-Menü Request Validator (Anforderungs-Validator) aus. Um die Anforderungsvalidierung zu deaktivieren, wählen Sie None (Keine) aus. Weitere Informationen zu den jeweiligen Optionen finden Sie unter [Validierung für REST APIs in API Gateway anfordern](#).
5. Wählen Sie API key required (API-Schlüssel erforderlich) aus, damit ein API-Schlüssel erforderlich ist. Wenn diese Option aktiviert ist, werden API-Schlüssel in [Nutzungsplänen](#) verwendet, um den Client-Datenverkehr zu drosseln.
6. (Optional) Um einen Operationsnamen in einem Java SDK dieser API zuzuweisen, die von einem API Gateway generiert wird, geben Sie unter Operation name (Operationsname) einen Namen ein. Beispielsweise ist für die Methodenanforderung GET `/pets/{petId}` der

entsprechende Java-SDK-Operationsname standardmäßig `GetPetsPetId`. Dieser Name wird aus dem HTTP-Verb (GET) der Methode und den Variablennamen des Ressourcenpfads (`Pets` und `PetId`) erstellt. Wenn Sie den Operationsnamen als `getPetById` festlegen, wird der SDK-Operationsnamen zu `GetPetById`.

7. Fügen Sie der Methode wie folgt einen Abfragezeichenfolge-Parameter hinzu:
 - a. Wählen Sie URL Query String Parameters (URL-Abfragezeichenfolgen-Parameter) und dann Add query string (Abfragezeichenfolge hinzufügen) aus.
 - b. Geben Sie unter Name den Namen des Abfragezeichenfolge-Parameters ein.
 - c. Wenn der neu erstellte Abfragezeichenfolge-Parameter für die Anforderungsvalidierung verwendet wird, wählen Sie die Option Required (Obligatorisch) aus. Weitere Informationen zur Anforderungsvalidierung finden Sie unter [Validierung für REST APIs in API Gateway anfordern](#).
 - d. Wenn der neu erstellte Abfragezeichenfolge-Parameter als Teil eines Caching-Schlüssels verwendet wird, wählen Sie die Option Caching aus. Weitere Informationen zum Caching finden Sie unter [Verwenden der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren](#).

Klicken Sie zum Entfernen des Abfragezeichenfolge-Parameters auf Remove (Entfernen).

8. Fügen Sie der Methode wie folgt einen Header-Parameter hinzu:
 - a. Wählen Sie HTTP Request Headers (HTTP-Anforderungs-Header) und dann Add header (Header hinzufügen) aus.
 - b. Geben Sie unter Name den Namen des Headers ein.
 - c. Wenn der neu erstellte Header für die Anforderungsvalidierung verwendet wird, wählen Sie die Option Required (Obligatorisch) aus. Weitere Informationen zur Anforderungsvalidierung finden Sie unter [Validierung für REST APIs in API Gateway anfordern](#).
 - d. Wenn der neu erstellte Header als Teil eines Caching-Schlüssels verwendet wird, wählen Sie die Option Caching aus. Weitere Informationen zum Caching finden Sie unter [Verwenden der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren](#).

Um den Header zu entfernen, wählen Sie Remove (Entfernen).

9. Um das Nutzlastformat einer Methodenanforderung mit dem POST-, PUT- oder PATCH-HTTP-Verb zu deklarieren, erweitern Sie Request Body (Anforderungstext) und führen Sie die folgenden Schritte aus:
 - a. Wählen Sie Add model aus.
 - b. Geben Sie unter Content-type einen MIME-Typ (beispielsweise application/json) ein.
 - c. Wählen Sie für Modell ein Modell aus dem Dropdown-Menü aus. Die derzeit verfügbaren Modelle für die API umfassen die standardmäßigen Empty- und Error-Modelle sowie alle Modelle, die Sie erstellt und der [Modell](#)-Sammlung der API hinzugefügt haben. Weitere Informationen zum Erstellen eines Modells finden Sie unter [Datenmodelle für REST APIs](#).

 Note

Das Modell ist nützlich, um den Client über das erwartete Datenformat einer Nutzlast zu informieren. Es ist hilfreich, ein Mapping-Vorlagenskelett zu generieren. Es ist wichtig, einen stark typisierten SDK der API in solchen Sprachen wie Java, C#, Objective-C und Swift zu generieren. Es ist nur erforderlich, wenn die Anforderungsvalidierung für die Nutzlast aktiviert ist.

10. Wählen Sie Speichern.

API Gateway-Methodenantwort über die API Gateway-Konsole einrichten

Eine API-Methode kann über eine oder mehrere Antworten verfügen. Jede Antwort wird von ihrem HTTP-Statuscode indiziert. Standardmäßig fügt die API Gateway-Konsole den Antworten der Methode eine 200-Antwort hinzu. Sie können dies ändern, z. B. so, dass die Methode stattdessen 201 zurückgibt. Sie können weitere Antworten hinzufügen, z. B. 409 bei Zugriffsverweigerungen und 500 bei Verwendungen nicht initialisierter Stufenvariablen.

Um die API Gateway-Konsole zum Ändern, Löschen oder Hinzufügen einer Antwort zu einer API-Methode zu verwenden, folgen Sie diesen Anweisungen.

1. Wählen Sie Ihre Methode im Bereich Ressourcen aus und klicken Sie dann auf die Registerkarte Methodenantwort. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie im Abschnitt Method response settings (Einstellungen für Methodenantwort) die Option Create response (Antwort erstellen) aus.

3. Geben Sie unter HTTP status code (HTTP-Statuscode) einen HTTP-Statuscode wie 200, 400 oder 500 ein.

Wenn für eine vom Backend zurückgegebene Antwort keine entsprechende Methodenantwort definiert ist, kann API Gateway die Antwort nicht an den Client zurückgeben. Stattdessen gibt es eine 500 Internal server error-Fehlerantwort zurück.

4. Wählen Sie Add header.
5. Geben Sie unter Header name (Header-Name) den Namen ein.

Um einen Header vom Backend an den Client zurückzugeben, fügen Sie den Header in die Methodenantwort ein.

6. Wählen Sie Add model (Modell hinzufügen), um ein Format für den Textkörpers der Methodenantwort zu definieren.

Geben Sie den Medientyp der Antwortnutzlast für Content type (Inhaltstyp) ein und wählen Sie ein Modell aus dem Dropdown-Menü Models (Modelle) aus.

7. Wählen Sie Speichern.

Um eine bestehende Antwort zu ändern, navigieren Sie zu Ihrer Methodenantwort und wählen Sie dann Edit (Bearbeiten). Um den HTTP status code (HTTP-Statuscode) zu ändern, wählen Sie Delete (Löschen) und erstellen Sie eine neue Methodenantwort.

Für jede vom Backend zurückgegebene Antwort, müssen Sie über eine kompatible Antwort verfügen, die als Methodenantwort konfiguriert ist. Jedoch ist die Konfiguration der Methodenantwort-Header und des Nutzlastmodells optional, es sei denn, Sie weisen das Ergebnis des Backends der Methodenantwort vor der Rückgabe an den Client zu. Zudem ist ein Nutzlastmodell der Methodenantwort wichtig, wenn Sie einen stark typisierten SDK für Ihre API generieren.

Steuern und verwalten Sie den Zugriff auf REST APIs in API Gateway

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre API.

Sie können die folgenden Mechanismen für die Authentifizierung und Autorisierung verwenden:

- Mit Ressourcenrichtlinien können Sie ressourcenbasierte Richtlinien erstellen, um den Zugriff auf Ihre APIs und Methoden von bestimmten Quell-IP-Adressen oder VPC-Endpunkten aus zu

erlauben oder zu verweigern. Weitere Informationen finden Sie unter [the section called “API Gateway-Ressourcenrichtlinien verwenden”](#).

- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen, die auf eine gesamte API oder einzelne Methoden angewendet werden können. Mithilfe von IAM-Rollen und -Richtlinien können Sie steuern, wer Ihre APIs erstellen und verwalten darf und wer sie aufrufen kann. Weitere Informationen finden Sie unter [the section called “Verwenden von IAM-Berechtigungen”](#).
- IAM-Tags können zusammen mit IAM-Richtlinien verwendet werden, um den Zugriff zu steuern. Weitere Informationen finden Sie unter [the section called “Attributbasierte Zugriffskontrolle”](#).
- Endpunktrichtlinien für Schnittstellen-VPC-Endpunkte [ermöglichen es Ihnen, IAM-Ressourcenrichtlinien an VPC-Schnittstellen-Endpunkte anzuhängen, um die Sicherheit Ihrer privaten Geräte zu verbessern. APIs](#) Weitere Informationen finden Sie unter [the section called “VPC-Endpunktrichtlinien für private Zwecke verwenden APIs”](#).
- Lambda-Genehmiger sind Lambda-Funktionen zur Steuerung des Zugriffs auf Ihre REST-API-Methoden unter Verwendung einer Bearer-Token-Authentifizierung sowie Informationen, die von Headern, Pfaden, Abfragezeichenfolgen, Stufenvariablen oder Kontextvariablen als Anfrageparameter beschrieben werden. Lambda-Genehmiger werden verwendet, um zu steuern, wer REST-API-Methoden aufrufen kann. Weitere Informationen finden Sie unter [the section called “Lambda-Genehmiger verwenden”](#).
- Mit Amazon Cognito Cognito-Benutzerpools können Sie anpassbare Authentifizierungs- und Autorisierungslösungen für Ihren REST APIs erstellen. Amazon Cognito-Benutzerpools werden verwendet, um zu steuern, wer REST-API-Methoden aufrufen kann. Weitere Informationen finden Sie unter [the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”](#).

Sie können die folgenden Mechanismen verwenden, um andere Aufgaben im Zusammenhang mit der Zugriffssteuerung auszuführen:

- Mittels Cross-Origin Resource Sharing (CORS) können Sie steuern, wie Ihre REST-API auf Cross-Domain-Ressourcenanforderungen reagiert. Weitere Informationen finden Sie unter [the section called “CORS”](#).
- Clientseitige SSL-Zertifikate können verwendet werden, um sicherzustellen, dass HTTP-Anfragen an Ihr Backend-System von API Gateway stammen. Weitere Informationen finden Sie unter [the section called “Clientzertifikate”](#).

- AWS WAF kann verwendet werden, um Ihre API-Gateway-API gegen häufige Web-Exploits zu schützen. Weitere Informationen finden Sie unter [the section called “AWS WAF”](#).

Sie können die folgenden Mechanismen verwenden, um den Zugriff nachzuverfolgen und einzuschränken, den Sie autorisierten Clients gewährt haben:

- Mit Nutzungsplänen können Sie API-Schlüssel für Ihre Kunden bereitstellen und die Verwendung Ihrer API-Stufen und Methoden für jeden API-Schlüssel nachverfolgen und beschränken. Weitere Informationen finden Sie unter [the section called “Nutzungspläne”](#).

Zugriff auf eine REST-API mit API Gateway-Ressourcenrichtlinien steuern

Amazon API Gateway-Ressourcenrichtlinien sind JSON-Richtliniendokumente, die Sie an eine API anhängen, um zu steuern, ob ein bestimmter Prinzipalbenutzer (normalerweise eine IAM-Rolle oder -Gruppe) die API aufrufen kann. Sie können API Gateway-Ressourcenrichtlinien verwenden, damit Ihre API sicher aufgerufen werden kann:

- Benutzer aus einem bestimmten AWS Konto.
- Angegebene Quell-IP-Adressbereiche oder CIDR-Blöcke.
- Spezifizierte virtuelle private Clouds (VPCs) oder VPC-Endpunkte (in einem beliebigen Konto).

Sie können eine Ressourcenrichtlinie für jeden API-Endpunkttyp in API Gateway anhängen AWS-Managementkonsole, indem Sie die AWS CLI oder verwenden AWS SDKs. Im [privaten Bereich APIs](#) können Sie Ressourcenrichtlinien zusammen mit VPC-Endpunktrichtlinien verwenden, um zu steuern, welche Prinzipale Zugriff auf welche Ressourcen und Aktionen haben. Weitere Informationen finden Sie unter [the section called “VPC-Endpunktrichtlinien für private Zwecke verwenden APIs”](#).

API-Gateway-Ressourcenrichtlinien unterscheiden sich von IAM-identitätsbasierten Richtlinien. IAM-identitätsbasierte Richtlinien sind IAM-Benutzern, -Gruppen oder -Rollen zugeordnet und definieren, welche Aktionen diese Identitäten mit welchen Ressourcen ausführen können. API Gateway-Ressourcenrichtlinien werden an Ressourcen angehängt. Sie können API Gateway-Ressourcenrichtlinien zusammen mit IAM-Richtlinien verwenden. Weitere Informationen finden Sie unter [Identitätsbasierte und ressourcenbasierte Richtlinien](#).

Themen

- [Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway](#)

- [So beeinflussen API Gateway-Ressourcenrichtlinien den Autorisierungs-Workflow](#)
- [Beispiele für API Gateway-Ressourcenrichtlinien](#)
- [API Gateway-Ressourcenrichtlinie erstellen und an eine API anhängen](#)
- [AWS Bedingungsschlüssel, die in API-Gateway-Ressourcenrichtlinien verwendet werden können](#)

Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway

Diese Seite beschreibt die grundlegenden Elemente, die in den Ressourcenrichtlinien für Amazon API Gateway verwendet werden.

Ressourcenrichtlinien werden mit derselben Syntax wie IAM-Richtlinien spezifiziert. Vollständige Informationen zur Richtliniensprache finden Sie unter [Übersicht der IAM-Richtlinien](#) und [AWS Identity and Access Management -Richtlinienverweis](#) im IAM-Benutzerhandbuch.

Informationen darüber, wie ein AWS Dienst entscheidet, ob eine bestimmte Anfrage zugelassen oder abgelehnt werden soll, finden Sie unter [Feststellen, ob eine Anfrage zugelassen oder verweigert wird](#).

Allgemeine Elemente einer Zugriffsrichtlinie

In ihrer einfachsten Form enthält eine Ressourcenrichtlinie die folgenden Elemente:

- Ressourcen — APIs sind die Amazon API Gateway Gateway-Ressourcen, für die Sie Berechtigungen zulassen oder verweigern können. In einer Richtlinie identifizieren Sie die Ressource mithilfe eines Amazon-Ressourcennamens (ARN). Sie können außerdem eine abgekürzte Syntax verwenden, die API Gateway beim Speichern einer Ressourcenrichtlinie automatisch auf den vollständigen ARN erweitert. Weitere Informationen hierzu finden Sie unter [Beispiele für API Gateway-Ressourcenrichtlinien](#).

Informationen zum Format des vollständigen Resource-Elements finden Sie unter [Ressourcenformat für Berechtigungen zur Ausführung der API in API Gateway](#).

- Aktionen – Für jede Ressource unterstützt Amazon API Gateway eine Reihe von Operationen. Sie identifizieren Ressourcenoperationen, die Sie zulassen (oder ablehnen) können, indem Sie Aktionsschlüsselwörter verwenden.

Beispielsweise kann der Benutzer mit der `execute-api:Invoke`-Berechtigung die API nach einer Clientanforderung aufrufen.

Informationen zum Format des Action-Elements finden Sie unter [Aktionsformat für Berechtigungen zur Ausführung der API in API Gateway](#).

- **Auswirkung** – Zeigt die Auswirkung, wenn der Benutzer die bestimmten Aktion anfordert – entweder Allow oder Deny. Sie können den Zugriff auf eine Ressource auch explizit verweigern. Damit können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.

Note

"Implizit verweigern" ist identisch mit "Standardmäßig verweigern".

Eine "implizite Verweigerung" unterscheidet sich von einer "expliziten Verweigerung".

Weitere Informationen finden Sie unter [Der Unterschied zwischen standardmäßiger und expliziter Zugriffsverweigerung](#).

- **Prinzipal** – das Konto oder der Benutzer, das oder der Zugriff auf die Aktionen und Ressourcen in der Anweisung hat. In einer Ressourcenrichtlinie ist der Prinzipal der Benutzer oder das Konto, der bzw. das die Berechtigung erhält.

Die folgende Beispiel-Ressourcenrichtlinie zeigt die zuvor genannten allgemeinen Richtlinienelemente. Die Richtlinie gewährt jedem Benutzer, dessen Quell-IP-Adresse sich *account-id* im Adressblock befindet *region*, Zugriff auf die API unter dem in der angegebenen Liste angegebenen Wert *123.4.5.6/24*. Die Richtlinie verweigert den Zugriff auf die API, wenn die Quell-IP des Benutzers nicht innerhalb des Bereichs liegt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111111111111:*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111111111111:*",
      "Condition": {
```

```
    "NotIpAddress": {  
      "aws:SourceIp": "123.4.5.6/24"  
    }  
  }  
}  
]  
}
```

So beeinflussen API Gateway-Ressourcenrichtlinien den Autorisierungs-Workflow

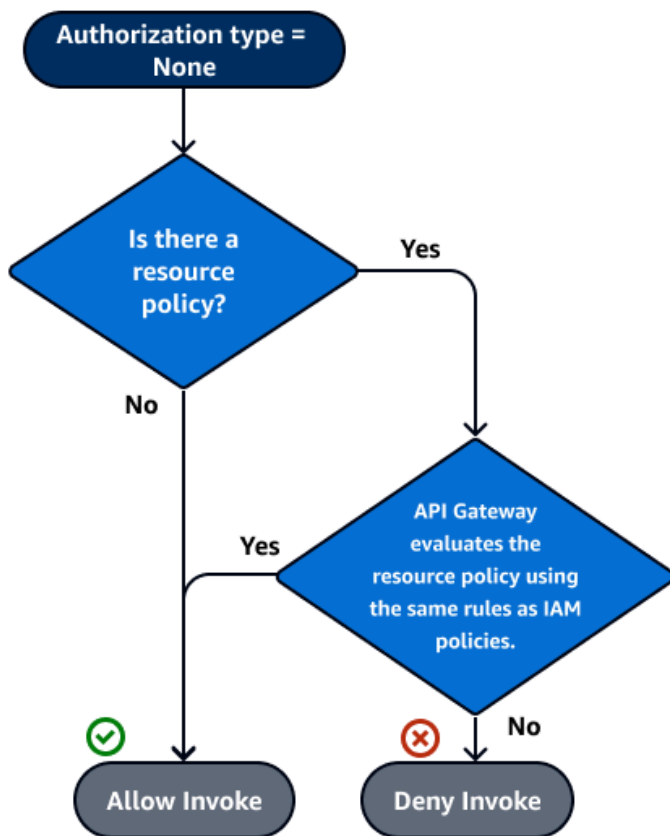
Wenn API Gateway die an die API angefügte Ressourcenrichtlinie ausgewertet wird, wird das Ergebnis vom Authentifizierungstyp beeinflusst, den Sie für die API definiert haben, wie in den Ablaufdiagrammen in den folgenden Abschnitten dargestellt.

Themen

- [Nur API Gateway-Ressourcenrichtlinie](#)
- [Lambda-Genehmiger und Ressourcenrichtlinie](#)
- [IAM-Authentifizierung und Ressourcenrichtlinie](#)
- [Amazon Cognito-Authentifizierung und Ressourcenrichtlinie](#)
- [Ergebnistabellen für die Richtlinienbewertung](#)

Nur API Gateway-Ressourcenrichtlinie

In diesem Workflow wird eine API Gateway-Ressourcenrichtlinie an die API angehängt. Es wird jedoch kein Authentifizierungstyp für die API definiert. Die Bewertung der Richtlinie beinhaltet das Suchen nach einer expliziten Erlaubnis basierend auf den eingehenden Kriterien des Aufrufers. Eine implizite Verweigerung oder eine explizite Verweigerung führt dazu, dass der Aufrufer abgelehnt wird.



Im Folgenden finden Sie ein Beispiel für eine solche Ressourcenrichtlinie.

JSON

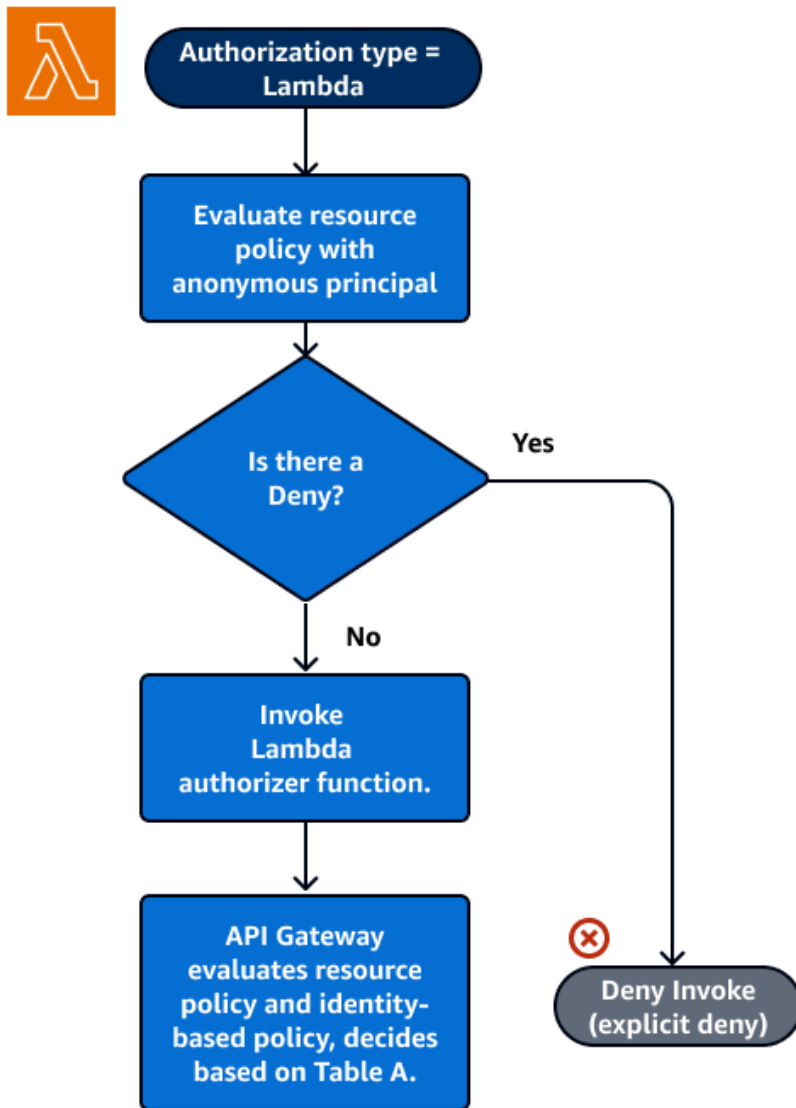
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
        }
      }
    }
  ]
}
```

```
}
```

Lambda-Genehmiger und Ressourcenrichtlinie

In diesem Workflow wird zusätzlich zu einer Ressourcenrichtlinie ein Lambda-Genehmiger für die API konfiguriert. Die Ressourcenrichtlinie wird in zwei Phasen ausgewertet. Bevor der Lambda-Genehmiger aufgerufen wird, evaluiert API Gateway zunächst die Richtlinie und prüft auf eine explizite Verweigerung. Wenn dies der Fall ist, wird dem Aufrufer der Zugriff sofort verweigert. Andernfalls wird der Lambda-Genehmiger aufgerufen, und es wird ein [Richtliniendokument](#) zurückgegeben, das in Verbindung mit der Ressourcenrichtlinie ausgewertet wird. Wenn Ihr Autorisierer Caching verwendet, gibt API Gateway möglicherweise das zwischengespeicherte Richtliniendokument zurück. Das Ergebnis wird basierend auf den Angaben in [Tabelle A](#) bestimmt.

Die folgende Beispielressourcenrichtlinie ermöglicht Aufrufe nur vom VPC-Endpunkt, dessen VPC-Endpunkt-ID lautet `vpce-1a2b3c4d`. Während der "Pre-Auth"-Evaluierung können nur die Aufrufe von dem im Beispiel angegebenen VPC-Endpunkt vorwärts gehen und den Lambda-Genehmiger auswerten. Alle verbleibenden Aufrufe werden blockiert. Dieser Autorisierungs-Workflow ist derselbe, wenn Sie einen benutzerdefinierten Domainnamen für eine private API verwenden.



JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/"
      ],
      "Condition": {

```

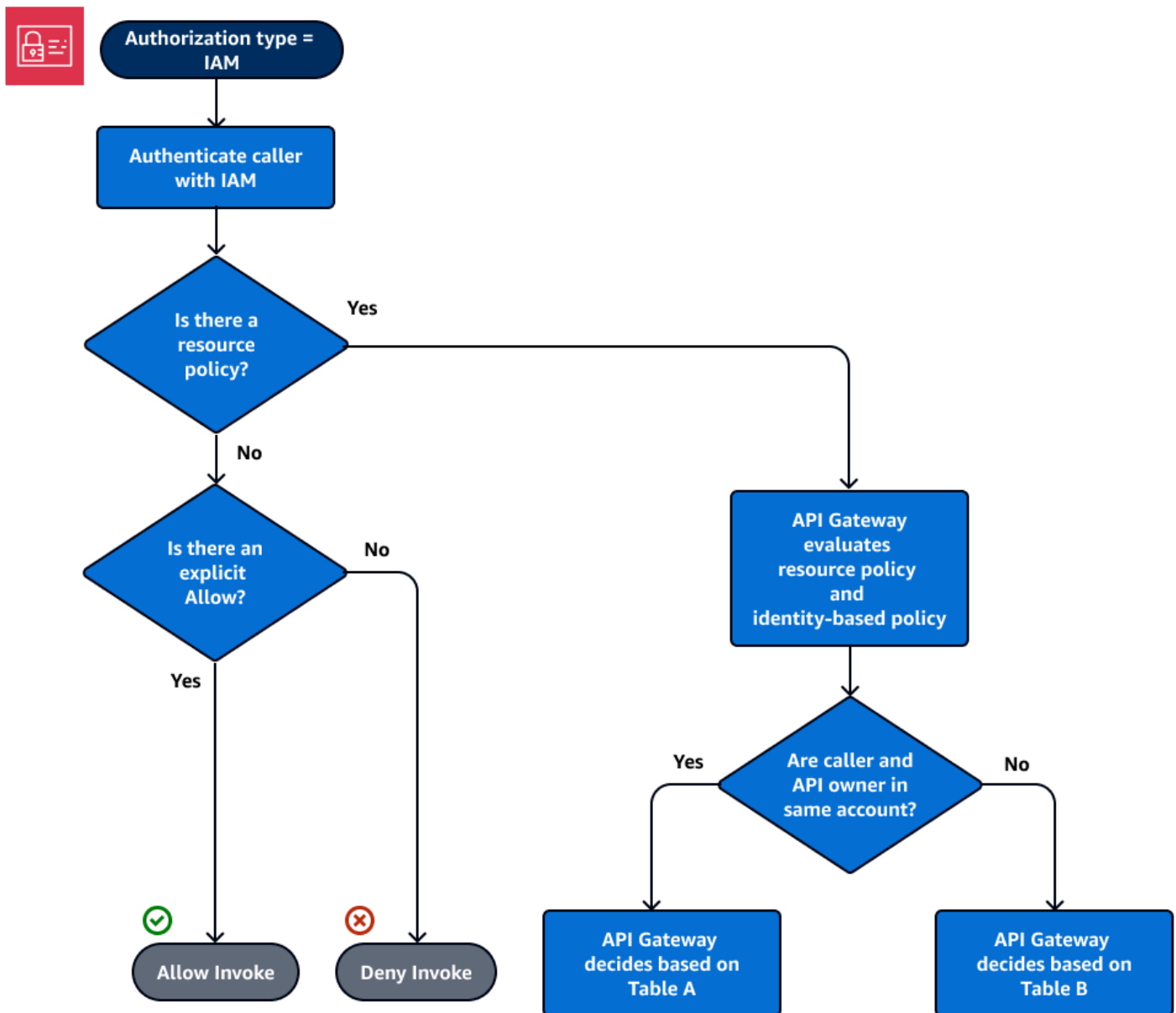
```
    "StringNotEquals": {  
      "aws:SourceVpce": "vpce-1a2b3c4d"  
    }  
  }  
]  
}
```

IAM-Authentifizierung und Ressourcenrichtlinie

In diesem Workflow wird die IAM-Authentifizierung für die API zusätzlich zu einer Ressourcenrichtlinie konfiguriert. Nach dem Authentifizieren des Benutzers mit dem IAM-Service wertet die API die dem Benutzer zugeordneten Richtlinien und die Ressourcenrichtlinie aus. Das Ergebnis hängt davon ab, ob sich der Anrufer im selben AWS-Konto oder einem anderen als AWS-Konto dem API-Besitzer befindet.

Wenn der Aufrufer und der API-Besitzer aus separaten Konten stammen, erlauben sowohl die IAM-Richtlinien als auch die Ressourcenrichtlinie dem Aufrufer explizit, fortzufahren. Weitere Informationen finden Sie unter [Tabelle B](#).

Wenn sich der Aufrufer und der API-Besitzer allerdings im selben AWS-Konto befinden, müssen die Benutzerrichtlinien oder die Ressourcenrichtlinie dem Aufrufer ausdrücklich erlauben, fortzufahren. Weitere Informationen finden Sie unter [Tabelle A](#).



Im Folgenden finden Sie ein Beispiel für eine kontoübergreifende Ressourcenrichtlinie. Unter der Annahme, dass die IAM-Richtlinie einen Zulassungseffekt enthält, erlaubt diese Ressourcenrichtlinie Aufrufe nur von der VPC, deren VPC-ID *vpc-2f09a348* ist. Weitere Informationen finden Sie unter [Tabelle B](#).

JSON

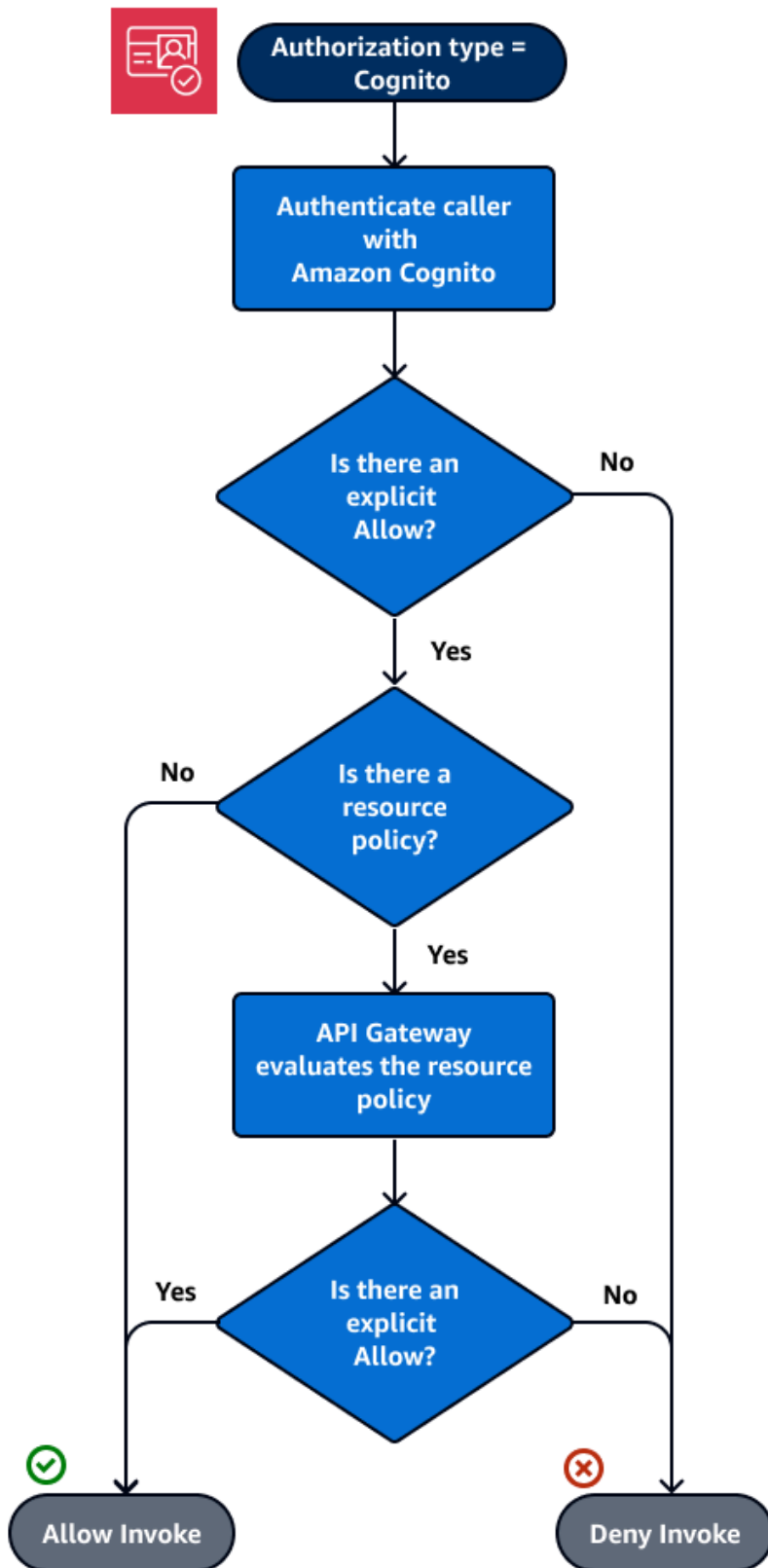
```
{
  "Version": "2012-10-17",
  "Statement": [
```



```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "execute-api:Invoke",
  "Resource": [
    "arn:aws:execute-api:us-east-1:111111111111:api-id/"
  ],
  "Condition": {
    "StringEquals": {
      "aws:SourceVpc": "vpc-2f09a348"
    }
  }
}
```

Amazon Cognito-Authentifizierung und Ressourcenrichtlinie

In diesem Workflow wird zusätzlich zu einer Ressourcenrichtlinie ein [Amazon Cognito-Benutzerpool](#) für die API konfiguriert. API Gateway versucht zunächst, den Aufrufer über Amazon Cognito zu authentifizieren. Dies wird in der Regel über ein [JWT-Token](#) durchgeführt, das vom Aufrufer bereitgestellt wird. Wenn die Authentifizierung erfolgreich ist, wird die Ressourcenrichtlinie unabhängig ausgewertet und eine explizite Genehmigung erforderlich. Eine Zugriffsverweigerung oder weder "Allow" (Zugriffserlaubnis) oder "Deny" (Zugriffsverweigerung) führt zu einer Zugriffsverweigerung. Es folgt ein Beispiel für eine Ressourcenrichtlinie, die zusammen mit Amazon Cognito-Benutzerpools verwendet werden könnte.



Im Folgenden finden Sie ein Beispiel für eine Ressourcenrichtlinie, die Aufrufe nur von einer bestimmten Quelle zulässt IPs, vorausgesetzt, dass das Amazon Cognito Cognito-Authentifizierungstoken eine Zulassung enthält. Weitere Informationen finden Sie unter [Tabelle B](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
        }
      }
    }
  ]
}
```

Ergebnistabellen für die Richtlinienbewertung

In Tabelle A wird das jeweilige Verhalten aufgeführt, wenn der Zugriff auf eine API-Gateway-API durch eine IAM-Richtlinie oder einen Lambda-Generierer und eine API-Gateway-Ressourcenrichtlinie gesteuert wird, die sich beide im selben AWS-Konto befinden.

IAM-Richtlinie (oder Lambda-Generierer)	API-Gateway-Ressourcenrichtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

IAM-Richtlinie (oder Lambda-Genehmiger)	API-Gateway-Ressourcenrichtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Deny	Explizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Weder Zulassen noch Verweigern	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Deny	Explizite Zugriffsverweigerung
Deny	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Explizite Zugriffsverweigerung
Deny	Weder Zulassen noch Verweigern	Explizite Zugriffsverweigerung
Deny	Deny	Explizite Zugriffsverweigerung

In Tabelle B ist das Verhalten aufgeführt, das entsteht, wenn der Zugriff auf eine API-Gateway-API durch eine IAM-Richtlinie gesteuert wird oder ein Amazon Cognito Cognito-Benutzerpool-Autorisierer und eine API Gateway Gateway-Ressourcenrichtlinie, die sich unterscheiden, gesteuert wird. AWS-Konten Wenn einer der beiden Parameter nicht zulässt (weder zulassen noch verweigern), wird der kontoübergreifende Zugriff verweigert. Ein kontoübergreifender Zugriff erfordert, dass sowohl die Ressourcenrichtlinie als auch die IAM-Richtlinie (oder ein Lambda- oder Amazon-Cognito-Benutzerpool-Genehmiger) den Zugriff ausdrücklich gewähren.

IAM-Richtlinie (oder Amazon-Cognito-Benutzerpool-Genehmiger)	API-Gateway-Ressourcenrichtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Deny	Explizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Deny	Explizite Zugriffsverweigerung
Deny	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Explizite Zugriffsverweigerung
Deny	Weder Zulassen noch Verweigern	Explizite Zugriffsverweigerung
Deny	Deny	Explizite Zugriffsverweigerung

Beispiele für API Gateway-Ressourcenrichtlinien

Auf dieser Seite werden einige Beispiele zu typischen Anwendungsfällen für API Gateway Ressourcenrichtlinien vorgestellt.

In den folgenden Beispielrichtlinien wird eine vereinfachte Syntax verwendet, um die API-Ressource anzugeben. Diese vereinfachte Syntax ist eine abgekürzte Form, mit der Sie auf eine API-Ressource verweisen können, anstatt den vollständigen Amazon-Ressourcenname (ARN) anzugeben. API Gateway konvertiert die abgekürzte Syntax in den vollständigen ARN, wenn Sie die Richtlinie speichern. Sie können zum Beispiel die Ressource `execute-api:/stage-name/GET/pets` in einer Ressourcenrichtlinie angeben. API Gateway wandelt die Ressource in `arn:aws:execute-api:us-east-2:123456789012:aabbccdde/stage-name/GET/pets` um, wenn Sie die Ressourcenrichtlinie speichern. API Gateway erstellt den vollständigen ARN unter Verwendung der aktuellen Region, Ihrer AWS Konto-ID und der ID der REST-API, mit der die Ressourcenrichtlinie verknüpft ist. Sie können mit `execute-api:/*` alle Phasen, Methoden und Pfade in der aktuellen API darstellen. Informationen zur Sprache der Zugriffsrichtlinie finden Sie unter [Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway](#).

Themen

- [Beispiel: Erlauben Sie Rollen in einem anderen AWS Konto, eine API zu verwenden](#)
- [Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs](#)
- [Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs bei Verwendung einer privaten API](#)
- [Beispiel: Erlauben von privatem API-Datenverkehr basierend auf der Quell-VPC oder dem VPC-Endpunkt](#)

Beispiel: Erlauben Sie Rollen in einem anderen AWS Konto, eine API zu verwenden

Die folgende Beispiel-Ressourcenrichtlinie gewährt API-Zugriff in einem AWS Konto auf zwei Rollen in einem anderen AWS Konto über die Protokolle [Signature Version 4](#) (Sigv4) oder [Signature Version 4a](#) (SigV4a). Insbesondere wird der Entwickler- und der Administratorrolle für das durch `account-id-2` identifizierte AWS Konto die `execute-api:Invoke` Aktion zur Ausführung der Aktion für die `GET pets` Ressource (API) in Ihrem Konto gewährt. AWS

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/developer",
          "arn:aws:iam::111122223333:role/Admin"
        ]
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*stage/GET/pets"
      ]
    }
  ]
}
```

Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs

Die folgende Beispiel-Ressourcenrichtlinie verweigert (blockiert) den eingehenden Datenverkehr zu einer API, wenn dieser von zwei bestimmten Quell-IP-Adressblöcken stammt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    }
  ],
}
```

```
{
  "Effect": "Deny",
  "Principal": "*",
  "Action": "execute-api:Invoke",
  "Resource": [
    "execute-api:/*"
  ],
  "Condition": {
    "IpAddress": {
      "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
    }
  }
}
```

Wenn Sie IAM-Benutzerrichtlinien oder API Gateway-Ressourcenrichtlinien verwenden, um den Zugriff auf API Gateway oder ein beliebiges API Gateway zu steuern APIs, stellen Sie sicher, dass Ihre Richtlinien aktualisiert wurden und IPv6 Adressbereiche enthalten. Richtlinien, die nicht für den Umgang mit IPv6 Adressen aktualisiert wurden, können sich auf den Zugriff von Clients auf API Gateway auswirken, wenn sie beginnen, den Dualstack-Endpunkt zu verwenden. Weitere Informationen finden Sie unter [the section called “IPv6 Adressen in IAM-Richtlinien verwenden”](#).

Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs bei Verwendung einer privaten API

Die folgende Beispiel-Ressourcenrichtlinie verweigert (blockiert) den eingehenden Datenverkehr zu einer privaten API, wenn dieser von zwei bestimmten Quell-IP-Adressblöcken stammt. Bei Verwendung von Private APIs `execute-api` schreibt der VPC-Endpunkt für die ursprüngliche Quell-IP-Adresse neu. Die `aws:VpcSourceIp`-Bedingung filtert die Anforderung anhand der ursprünglichen IP-Adresse des Anforderers.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
```



```

        "Action": "execute-api:Invoke",
        "Resource": [
            "execute-api/*"
        ]
    },
    {
        "Effect": "Deny",
        "Principal": "*",
        "Action": "execute-api:Invoke",
        "Resource": [
            "execute-api/*"
        ],
        "Condition": {
            "IpAddress": {
                "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
            }
        }
    }
}

```

Beispiel: Erlauben von privatem API-Datenverkehr basierend auf der Quell-VPC oder dem VPC-Endpoint

Die folgenden Beispiel-Ressourcenrichtlinien erlauben eingehenden Datenverkehr zu einer privaten API von nur einer bestimmten Virtual Private Cloud (VPC) oder einem bestimmten VPC-Endpoint.

In diesem Beispiel für eine Ressourcenrichtlinie wird eine Quell-VPC angegeben:

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api/*"
            ]
        }
    ],
}

```

```

    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ],
      "Condition" : {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-1a2b3c4d"
        }
      }
    }
  ]
}

```

In diesem Beispiel für eine Ressourcenrichtlinie wird ein Quell-VPC-Endpunkt angegeben:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ],
      "Condition" : {
        "StringNotEquals": {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
      }
    }
  ]
}

```

```
}  
  }  
    ]  
  }
```

API Gateway-Ressourcenrichtlinie erstellen und an eine API anhängen

Sie müssen eine API-Gateway-Ressourcenrichtlinie erstellen und diese an die API anhängen, um einem Benutzer den Zugriff auf Ihre API durch Aufruf des API-Ausführungsdienstes zu ermöglichen. Wenn Sie eine Richtlinie mit Ihrer API verknüpfen werden die Berechtigungen in der Richtlinie auf die API-Methoden angewendet. Wenn Sie die Ressourcenrichtlinie aktualisieren, müssen Sie die API erneut bereitstellen.

Themen

- [Voraussetzungen](#)
- [Verknüpfen einer Ressourcenrichtlinie mit einer API-Gateway-API](#)
- [Fehlerbehebung bei Ressourcenrichtlinien](#)

Voraussetzungen

Für die Aktualisierung einer API-Gateway-Ressourcenrichtlinie sind die Berechtigungen `apigateway:PATCH` und `apigateway:UpdateRestApiPolicy` erforderlich.

Bei einer Edge-optimierten oder regionalen API können Sie Ihre Ressourcenrichtlinie bereits bei der Erstellung an Ihre API anhängen, oder nachdem sie bereitgestellt wurde. Eine private API kann nicht ohne eine Ressourcenrichtlinie bereitgestellt werden. Weitere Informationen finden Sie unter [the section called “Privates REST APIs”](#).

Verknüpfen einer Ressourcenrichtlinie mit einer API-Gateway-API

Im folgenden Verfahren sehen Sie, wie Sie eine Ressourcenrichtlinie an eine API-Gateway-API anfügen.

AWS-Managementkonsole

So hängen Sie eine Ressourcenrichtlinie an eine API Gateway-API an:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.

2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Wählen Sie Richtlinie erstellen aus.
5. (Optional) Wählen Sie Vorlage auswählen, um eine Beispielrichtlinie zu generieren.

In den Beispielrichtlinien sind Platzhalter in doppelte geschweifte Klammern ("{{*placeholder*}}") eingeschlossen. Ersetzen Sie die einzelnen Platzhalter (einschließlich der geschweiften Klammern) durch die erforderlichen Informationen.

6. Wenn Sie keines der Vorlagenbeispiele verwenden, geben Sie Ihre Ressourcenrichtlinie ein.
7. Wählen Sie Änderungen speichern aus.

Wenn die API zuvor in der API Gateway-Konsole bereitgestellt wurde, müssen Sie sie neu bereitstellen, damit die Ressourcenrichtlinie wirksam wird.

AWS CLI

Verwenden Sie den AWS CLI folgenden Befehl, um eine neue API zu erstellen und ihr eine Ressourcenrichtlinie anzuhängen: [create-rest-api](#)

```
aws apigateway create-rest-api \
  --name "api-name" \
  --policy "{\"jsonEscapedPolicyDocument\"}"
```

Verwenden Sie den AWS CLI folgenden [update-rest-api](#) Befehl, um eine Ressourcenrichtlinie an eine bestehende API anzuhängen:

```
aws apigateway update-rest-api \
  --rest-api-id api-id \
  --patch-operations op=replace,path=/
policy,value='\"jsonEscapedPolicyDocument\"'
```

Sie können Ihre Ressourcenrichtlinie auch als separate `policy.json` Datei anhängen und sie in Ihren [create-rest-api](#) Befehl aufnehmen. Der folgende [create-rest-api](#) Befehl erstellt eine neue API mit einer Ressourcenrichtlinie:

```
aws apigateway create-rest-api \
  --name "api-name" \
```

```
--policy file://policy.json
```

`policy.json` ist eine API-Gateway-Ressourcenrichtlinie wie [the section called “Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs”](#).

AWS CloudFormation

Sie können CloudFormation es verwenden, um eine API mit einer Ressourcenrichtlinie zu erstellen. Im folgenden Beispiel wird eine REST-API mit der Beispiel-Ressourcenrichtlinie [the section called “Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs”](#) erstellt.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api/*'
        Condition:
          IpAddress:
            'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
      Version: 2012-10-17
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
```

```

    HttpMethod: GET
    ApiKeyRequired: false
    AuthorizationType: NONE
    Integration:
      Type: MOCK
      RequestTemplates:
        application/json: '{"statusCode": 200}'
      IntegrationResponses:
        - StatusCode: 200
          ResponseTemplates:
            application/json: '{}'
      MethodResponses:
        - StatusCode: 200
          ResponseModels:
            application/json: 'Empty'
    ApiDeployment:
      Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - MethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: test

```

Fehlerbehebung bei Ressourcenrichtlinien

Die folgenden Anleitungen zur Fehlerbehebung können Ihnen helfen, Probleme mit Ihrer Ressourcenrichtlinie zu lösen.

Meine API gibt {"Message": "User: anonymous is not authorized to perform: execute-api:invoke on resource: arn:aws:execute-api:us-east-1: *****/****/****/" } zurück

Wenn Sie in Ihrer Ressourcenrichtlinie den Principal auf einen AWS Prinzipal festlegen, wie zum Beispiel:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {

```

```
        "AWS": [
            "arn:aws:iam::111111111111:role/developer",
            "arn:aws:iam::111111111111:role/Admin"
        ],
        "Action": "execute-api:Invoke",
        "Resource": [
            "execute-api:/stage/GET/pets"
        ]
    }
}
```

Dann müssen Sie die AWS_IAM-Genehmigung für jede einzelne Methode in Ihrer API verwenden, andernfalls gibt Ihre API die o. g. Fehlermeldung zurück. Weitere Anweisungen zum Aktivieren der AWS_IAM-Genehmigung für eine Methode finden Sie unter [the section called “Methoden”](#).

Meine Ressourcenrichtlinie aktualisiert nicht

Wenn Sie die Ressourcenrichtlinie nach dem Erstellen der API aktualisieren, müssen Sie die API bereitstellen, damit die Änderungen übernommen werden, nachdem Sie die aktualisierte Richtlinie angefügt haben. Das Aktualisieren oder Speichern der Richtlinie allein ändert das Laufzeitverhalten der API nicht. Weitere Informationen zum Bereitstellen Ihrer API finden Sie unter [the section called “REST-APIs bereitstellen”](#).

Meine Ressourcenrichtlinie gibt den folgenden Fehler zurück: „Invalid policy document“ (Ungültiges Richtliniendokument). Bitte überprüfen Sie die Richtliniensyntax und stellen Sie sicher, dass die Prinzipale gültig sind.

Als ersten Schritt für die Fehlerbehebung empfehlen wir eine Überprüfung der Richtliniensyntax. Weitere Informationen finden Sie unter [the section called “Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway”](#). Außerdem empfehlen wir eine Überprüfung der Gültigkeit aller angegebenen Prinzipale und dass diese nicht gelöscht wurden.

Wenn sich Ihre API in einer [Opt-in-Region](#) befindet, sollten Sie weiterhin sicherstellen, dass die Region für alle Konten in der Ressourcenrichtlinie aktiviert ist.

AWS Bedingungsschlüssel, die in API-Gateway-Ressourcenrichtlinien verwendet werden können

Die folgende Tabelle enthält AWS Bedingungsschlüssel, die in Ressourcenrichtlinien oder APIs in API Gateway für jeden Autorisierungstyp verwendet werden können.

Weitere Informationen zu AWS Bedingungsschlüsseln finden Sie unter [AWS Globale Bedingungskontextschlüssel](#).

Bedingungsschlüssel	Kriterien	Erfordert AuthN ?	Genehmigungstyp
<code>aws:CurrentTime</code>	Keine	Nein	Alle
<code>aws:EpochTime</code>	Keine	Nein	Alle
<code>aws:Token IssueTime</code>	Schlüssel ist nur in Anforderungen vorhanden, die mithilfe temporärer Sicherheitsanmeldeinformationen signiert werden.	Ja	IAM
<code>aws:Multi FactorAuth Present</code>	Schlüssel ist nur in Anforderungen vorhanden, die mithilfe temporärer Sicherheitsanmeldeinformationen signiert werden.	Ja	IAM
<code>aws:Multi FactorAuthAge</code>	Schlüssel ist nur vorhanden, wenn die MFA in den Anforderungen enthalten ist.	Ja	IAM
<code>aws:Princ ipalAccount</code>	Keine	Ja	IAM
<code>aws:Princ ipalArn</code>	Keine	Ja	IAM
<code>aws:Princ ipalOrgID</code>	Dieser Schlüssel ist nur dann im Anforderungskontext	Ja	IAM

Bedingungsschlüssel	Kriterien	Erfordert AuthN ?	Genehmigungstyp
<code>aws:PrincipalOrgPaths</code>	Dieser Schlüssel ist nur dann im Anforderungskontext enthalten, wenn der Prinzipal Mitglied einer Organisation ist.	Ja	IAM
<code>aws:PrincipalTag</code>	Dieser Schlüssel ist nur dann im Anforderungskontext enthalten, wenn der Prinzipal ein IAM-Benutzer mit angefügten Tags ist. Er ist für einen Prinzipal enthalten, der eine IAM-Rolle mit angefügten Tags oder Sitzungs-Tags verwendet.	Ja	IAM
<code>aws:PrincipalType</code>	Keine	Ja	IAM
<code>aws:Referer</code>	Schlüssel ist nur vorhanden, wenn der Wert vom Aufrufer im HTTP-Header bereitgestellt wird.	Nein	Alle
<code>aws:SecureTransport</code>	Keine	Nein	Alle

Bedingungsschlüssel	Kriterien	Erfordert AuthN ?	Genehmigungstyp
<code>aws:SourceArn</code>	Keine	Nein	Alle
<code>aws:SourceIp</code>	Keine	Nein	Alle
<code>aws:SourceVpc</code>	Dieser Schlüssel kann nur für private Zwecke verwendet werden APIs.	Nein	Alle
<code>aws:SourceVpce</code>	Dieser Schlüssel kann nur für private Zwecke verwendet werden APIs.	Nein	Alle
<code>aws:VpcSourceIp</code>	Dieser Schlüssel kann nur für private Zwecke verwendet werden APIs.	Nein	Alle
<code>aws:UserAgent</code>	Schlüssel ist nur vorhanden, wenn der Wert vom Aufrufer im HTTP-Header bereitgestellt wird.	Nein	Alle
<code>aws:userid</code>	Keine	Ja	IAM
<code>aws:username</code>	Keine	Ja	IAM

Zugriffskontrolle für eine API mit IAM-Berechtigungen

Sie steuern den Zugriff auf Ihre Amazon API Gateway-API über [IAM-Berechtigungen](#), indem Sie den Zugriff auf die folgenden zwei API Gateway-Komponentenprozesse steuern:

- Um eine API in API Gateway zu erstellen, bereitzustellen und zu verwalten, müssen Sie dem API-Entwickler Berechtigungen zur Durchführung der erforderlichen Aktionen erteilen, die von der API-Verwaltungskomponente von API Gateway unterstützt werden.
- Zum Aufrufen einer bereitgestellten API oder zum Aktualisieren der API-Zwischenspeicherung benötigt der API-Aufrufer die Berechtigungen für die erforderlichen IAM-Aktionen, die von der API-Ausführungskomponente von Amazon API Gateway unterstützt werden.

Die Zugriffskontrolle für die beiden Prozesse umfasst verschiedene Berechtigungsmodelle, die nachstehend erläutert werden.

API Gateway-Berechtigungsmodell für die Erstellung und Verwaltung einer API

Wenn Sie einem API-Entwickler ermöglichen möchten, eine API in Amazon API Gateway zu erstellen und zu verwalten, müssen Sie [IAM-Berechtigungsrichtlinien erstellen](#), die zulassen, dass ein bestimmter API-Entwickler die erforderlichen [API-Entitäten](#) erstellen, aktualisieren, bereitstellen, ansehen oder löschen darf. Sie fügen die Richtlinie an einen Benutzer, eine Rolle oder eine Gruppe an.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Weitere Informationen dazu, wie Sie dieses Berechtigungsmodell verwenden, finden Sie unter [the section called "Identitätsbasierte API-Gateway-Richtlinien"](#).

API Gateway-Berechtigungsmodell für den Aufruf einer API

Wenn Sie zulassen möchten, dass ein API-Aufrufer die API aufruft oder deren Cache aktualisiert, müssen Sie IAM-Richtlinien erstellen, die einen bestimmten API-Aufrufer die API-Methode aufrufen lassen, für die die Benutzerauthentifizierung aktiviert ist. Der API-Entwickler legt `authorizationType` als `AWS_IAM`-Eigenschaft der Methode fest, damit der Aufrufer gezwungen ist, für die Authentifizierung die Anmeldeinformationen des Benutzers zu übermitteln. API Gateway unterstützt Signature Version 4a (SigV4a) und Signature Version 4 (SigV4) zur Authentifizierung der Benutzeranmeldeinformationen. Weitere Informationen finden Sie unter [AWS Signature Version 4](#). Anschließend verknüpfen Sie die Richtlinie mit einem Benutzer, einer Rolle oder einer Gruppe.

In dieser IAM-Berechtigungsrichtlinienanweisung enthält das IAM-Resource-Element eine Liste mit bereitgestellten API-Methoden, die durch bestimmte HTTP-Verben und API Gateway-[Ressourcenpfade](#) identifiziert werden. Das IAM Action-Element enthält die erforderliche API Gateway-API, die Aktionen ausführt. Zu diesen Aktionen gehören `execute-api:Invoke` oder `execute-api:InvalidCache`, wobei `execute-api` die zugrundeliegende API-Ausführungskomponente von API Gateway bezeichnet.

Weitere Informationen dazu, wie Sie dieses Berechtigungsmodell verwenden, finden Sie unter [Kontrollieren des Zugriffs für den API-Aufruf](#).

Wenn eine API in einen AWS Dienst (z. B. AWS Lambda) im Backend integriert ist, muss API Gateway auch über Berechtigungen für den Zugriff auf integrierte AWS Ressourcen (z. B. das Aufrufen einer Lambda-Funktion) im Namen des API-Aufrufers verfügen. Um diese Berechtigungen zu erteilen, erstellen Sie eine IAM-Rolle vom Typ AWS -Service für API Gateway. Wenn Sie diese Rolle in der IAM-Managementkonsole erstellen, enthält sie die folgende IAM-Vertrauensrichtlinie. In dieser wird Amazon API Gateway als vertrauenswürdige Entity deklariert, der die Übernahme der Rolle gestattet ist:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      }
    }
  ]
}
```

```
    },  
    "Action": "sts:AssumeRole"  
  }  
]  
}
```

Wenn Sie die IAM-Rolle durch Aufruf des Befehls [create-role](#) der CLI oder einer entsprechenden SDK-Methode erstellen, müssen Sie die oben beschriebene Vertrauensrichtlinie als Eingabeparameter von `assume-role-policy-document` bereitstellen. Versuchen Sie nicht, eine solche Richtlinie direkt in der IAM-Managementkonsole zu erstellen oder den Befehl AWS CLI [create-policy](#) oder eine entsprechende SDK-Methode aufzurufen.

Damit API Gateway den integrierten AWS Dienst aufrufen kann, müssen Sie dieser Rolle auch entsprechende IAM-Berechtigungsrichtlinien für den Aufruf integrierter AWS Dienste zuordnen. Um beispielsweise eine Lambda-Funktion aufrufen zu können, müssen Sie die folgende IAM-Berechtigungsrichtlinie in die IAM-Rolle einfügen:

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "lambda:InvokeFunction",  
      "Resource": "*"   
    }  
  ]  
}
```

Lambda unterstützt ressourcenbasierte Zugriffsrichtlinien und kombiniert somit Vertrauens- und Berechtigungsrichtlinien. Bei der Integration einer API mit einer Lambda-Funktion über die API Gateway-Konsole werden Sie nicht aufgefordert, diese IAM-Rolle explizit festzulegen, da die Konsole die ressourcenbasierten Berechtigungen für die Lambda-Funktion mit Ihrer Zustimmung für Sie festlegt.

 Note

Um die Zugriffskontrolle für einen AWS Dienst festzulegen, können Sie entweder das aufruferbasierte Berechtigungsmodell verwenden, bei dem eine Berechtigungsrichtlinie direkt an den Benutzer oder die Gruppe des Anrufers angehängt wird, oder das rollenbasierte Berechtigungsmodell, bei dem eine Berechtigungsrichtlinie an eine IAM-Rolle angehängt wird, die API Gateway übernehmen kann. Die Berechtigungsrichtlinien in den zwei Modellen können sich unterscheiden. Zum Beispiel kann die aufruferbasierte Richtlinie den Zugriff blockieren, während die rollenbasierte ihn zulässt. Sie können dies nutzen, um zu verlangen, dass ein Benutzer nur über eine API Gateway auf einen AWS Dienst zugreift.

Kontrollieren des Zugriffs für den API-Aufruf

In diesem Abschnitt erfahren Sie mehr über das Berechtigungsmodell für die Zugriffskontrolle Ihrer API mithilfe von IAM-Berechtigungen. Wenn die IAM-Autorisierung aktiviert ist, müssen Clients Signature Version 4a (SigV4a) und Signature Version 4 (Sigv4) verwenden, um ihre Anfragen mit Anmeldeinformationen zu signieren. AWS Weitere Informationen finden Sie unter [AWS Signature Version 4](#).

In diesem Abschnitt finden Sie eine Vorlage einer IAM-Richtlinienanweisung sowie die Referenz zur Richtlinienanweisung. In der Richtlinienanweisungsreferenz sind die Formate der Action- und Resource-Felder für den API-Ausführungsservice enthalten. Mithilfe dieser Referenzen erstellen Sie Ihre IAM-Richtlinienanweisung. Bei der Erstellung Ihrer IAM-Richtlinienanweisung sollten Sie auch berücksichtigen, wie sich Ressourcenrichtlinien von API Gateway auf den Genehmigungsworkflow auswirken. Weitere Informationen finden Sie unter [the section called “Wie Ressourcenrichtlinien den Autorisierungsworkflow beeinflussen”](#).

Für private Zwecke APIs sollten Sie eine Kombination aus einer API-Gateway-Ressourcenrichtlinie und einer VPC-Endpunktrichtlinie verwenden. Weitere Informationen finden Sie unter den folgenden Themen:

- [the section called “API Gateway-Ressourcenrichtlinien verwenden”](#)
- [the section called “VPC-Endpunktrichtlinien für private Zwecke verwenden APIs”](#)

Mit IAM-Richtlinien steuern, wer eine API Gateway-API-Methode aufrufen kann

Um mit IAM-Berechtigungen zu steuern, wer eine bereitgestellte API aufrufen kann oder nicht, erstellen Sie ein IAM-Richtliniendokument mit den erforderlichen Berechtigungen. Eine Vorlage für ein solches Richtliniendokument sehen Sie im Folgenden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Permission",
      "Action": [
        "execute-api:Execution-operation"
      ],
      "Resource": [
        "arn:aws:execute-api:region:123456789012:api-id/stage/METHOD_HTTP_VERB/Resource-path"
      ]
    }
  ]
}
```

Hier muss *Permission* durch Allow oder Deny ersetzt werden, abhängig davon, ob Sie die enthaltenen Berechtigungen erteilen oder entziehen möchten. *Execution-operation* ist durch die vom API-Ausführungsdienst unterstützten Operationen zu ersetzen. *METHOD_HTTP_VERB* steht für ein HTTP-Verb, das von den angegebenen Ressourcen unterstützt wird. *Resource-path* ist der Platzhalter für den URL-Pfad einer bereitgestellten API-[Resource](#)-Instance, die das genannte *METHOD_HTTP_VERB* unterstützt. Weitere Informationen finden Sie unter [Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway](#).

 Note

Damit IAM-Richtlinien wirksam sind, müssen Sie die IAM-Authentifizierung bei API-Methoden aktiviert haben, indem Sie AWS_IAM für die Eigenschaft [authorizationType](#) der Methoden festlegen. Andernfalls werden diese API-Methoden öffentlich zugänglich gemacht.

Angenommen, Sie möchten einem Benutzer die Berechtigung erteilen, eine Liste mit Haustieren anzusehen, die von einer bestimmten API zur Verfügung gestellt wird. Der Benutzer darf der Liste aber kein Haustier hinzufügen. Für diesen Fall können Sie die folgende Anweisung in die IAM-Richtlinie einschließen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/GET/pets"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/POST/pets"
      ]
    }
  ]
}
```

Um einem Benutzer die Berechtigung zum Anzeigen eines bestimmten Haustiers zu erteilen, das von einer API zur Verfügung gestellt wird, die als GET /pets/{petId} konfiguriert ist, können Sie die folgende Anweisung in die IAM-Richtlinie einschließen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```

    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111122223333:api-id/*/
GET/pets/a1b2"
      ]
    }
  ]
}

```

Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway

Im Folgenden werden die "Action"- und "Resource"-Formate von IAM-Richtlinienanweisungen für Zugriffsberechtigungen zur Ausführung einer API beschrieben.

Aktionsformat für Berechtigungen zur Ausführung der API in API Gateway

Der Action-Ausdruck zur API-Ausführung weist das folgende allgemeine Format auf:

```
execute-api:action
```

wo *action* ist eine API-ausführende Aktion verfügbar:

- *, repräsentiert alle der folgenden Aktionen.
- Invoke – wird verwendet, um eine API für eine Client-Anforderung aufzurufen.
- InvalidateCache – um den API-Cache für eine Client-Anforderung ungültig zu machen.

Ressourcenformat für Berechtigungen zur Ausführung der API in API Gateway

Der Resource-Ausdruck zur API-Ausführung weist das folgende allgemeine Format auf:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-specifier
```

Wobei:

- **region** ist die AWS Region (z. B. **us-east-1** oder ***** für alle AWS Regionen), die der bereitgestellten API für die Methode entspricht.
- **account-id** ist die 12-stellige AWS Konto-ID des REST-API-Besitzers.
- **api-id** ist der Identifier, den API Gateway der API für die Methode zugewiesen hat.
- **stage-name** ist der Name der Phase, die der Methode zugeordnet ist.
- **HTTP-VERB** ist das HTTP-Verb für die Methode. Es kann eines der folgenden Verben sein: GET, POST, PUT, DELETE, PATCH.
- **resource-path-specifier** ist der Pfad zur gewünschten Methode.

 Note

Wenn Sie einen Platzhalter (*) angeben, wendet der Resource-Ausdruck den Platzhalter auf den Rest des Ausdrucks an.

Hier einige Ressourcenausdrücke als Beispiel:

- **arn:aws:execute-api:::*:** für jeden Ressourcenpfad in jeder Phase, für jede API in jeder AWS Region.
- **arn:aws:execute-api:us-east-1:::*:** für jeden Ressourcenpfad in jeder Phase, für jede API in der AWS Region von us-east-1.
- **arn:aws:execute-api:us-east-1:::**api-id**/*** für jeden Ressourcenpfad in jeder Phase, für die API mit der Kennung **api-id** in der AWS Region us-east-1.
- **arn:aws:execute-api:us-east-1:::**api-id**/test/*** für jeden Ressourcenpfad in der Phase von test, für die API mit der Kennung von **api-id** in der AWS Region us-east-1.

Weitere Informationen hierzu finden Sie unter [Referenz zu API Gateway Amazon-Ressourcenname \(ARN\)](#).

IAM-Richtlinienbeispiele für API-Ausführungsberechtigungen

Weitere Informationen zu Berechtigungsmodellen und mehr Hintergrundwissen erhalten Sie unter [Kontrollieren des Zugriffs für den API-Aufruf](#).

Mit der folgenden Richtlinienanweisung erhält der Benutzer die Berechtigung zum Aufrufen jeder POST-Methode entlang dem Pfad von mydemoresource in der test-Stufe für die API mit der ID

a123456789. Dabei wird vorausgesetzt, dass die entsprechende API in der AWS -Region us-east-1 bereitgestellt wurde:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/my-demo-resource-
path/*"
      ]
    }
  ]
}
```

Die folgende Beispielrichtlinienanweisung erteilt dem Benutzer die Berechtigung zum Aufrufen jeder Methode im Ressourcenpfad von petstorewalkthrough/pets in allen Stufen, für die API mit der ID a123456789, in jeder AWS -Region, in der die entsprechende API bereitgestellt wurde:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

}

Verwenden Sie VPC-Endpunktrichtlinien für private Zwecke APIs in API Gateway

Sie können Sie eine VPC-Endpunktrichtlinie erstellen, um die Sicherheit Ihrer privaten API zu verbessern. Eine VPC-Endpunktrichtlinie ist eine IAM-Ressourcenrichtlinie, die Sie Ihrem VPC-Endpunkt anfügen können. Weitere Informationen finden Sie unter [Steuern des Zugriffs auf Services mit VPC-Endpunkten](#).

Eine VPC-Endpunktrichtlinie für die folgenden Aufgaben wäre zu empfehlen.

- Beschränken Sie den Zugriff auf Ihren VPC-Endpunkt und den Aufruf Ihrer API auf bestimmte Organisationen und Ressourcen.
- Verwenden Sie eine einzige Richtlinie und vermeiden Sie Sitzungs- oder rollenbasierte Richtlinien, um den Datenverkehr zu Ihrer API zu kontrollieren.
- Verschärfen Sie den Sicherheitsbereich Ihrer Anwendung bei der Migration von lokal zu AWS

Erwägungen zur VPC-Endpunktrichtlinie

Im Folgenden finden Sie Überlegungen zu Ihrer VPC-Endpunktrichtlinie.

- Die Identität des Aufrufers wird anhand des `Authorization-Header`-Werts bewertet. Zunächst wird die VPC-Endpunktrichtlinie ausgewertet, anschließend wertet API Gateway die Anfrage anhand der für die Methodenanforderung konfigurierten Autorisierungstypen aus. Die folgende Tabelle zeigt, wie die VPC-Endpunktrichtlinie anhand des Inhalts des `Authorization-Header`-Werts ausgewertet wird.

Inhalt des Authorization -Header-Werts	Auswertung der VPC-Endpunkt-Richtlinie
Kein Inhalt	Der Aufrufer wird als anonym Benutzer bewertet
Gültige SigV4- oder SigV4a -Signatur	Der Aufrufer wird anhand der Signatur als authentifizierte IAM-Identität bewertet
Ungültige SigV4- oder SigV4a -Signatur	API Gateway verweigert den Zugriff

Inhalt des Authorization -Header-Werts	Auswertung der VPC-Endpunkt-Richtlinie
Nicht-SigV4-Autorisierungsinformationen, z. B. ein Bearer-Token	Der Aufrufer wird als anonym Benutzer bewertet

- Wenn Ihre Zugriffskontrolle auf der Verwendung eines Bearer-Tokens basiert, wie beispielsweise einem Lambda- oder Amazon Cognito-Genehmiger, können Sie Ihren Sicherheitsbereich mithilfe der [Eigenschaften der Ressource](#) steuern.
- Wenn Ihre Autorisierungskontrollen die IAM-Autorisierung verwenden, können Sie Ihren Sicherheitsbereich mithilfe der [Eigenschaften der Ressource](#) und der [Eigenschaften des Prinzipals](#) steuern.
- VPC-Endpunktrichtlinien können zusammen mit API Gateway-Ressourcenrichtlinien verwendet werden. Die API-Gateway-Ressourcenrichtlinie legt fest, welche Prinzipale auf die API zugreifen können. Die Endpunktrichtlinie legt fest, wer auf die VPC zugreifen APIs kann und welche vom VPC-Endpunkt aus aufgerufen werden können. Für Ihre private API ist eine Ressourcenrichtlinie erforderlich, Sie müssen jedoch keine benutzerdefinierte VPC-Endpunktrichtlinie erstellen.

Beispiele für VPC-Endpunktrichtlinien

Sie können Richtlinien mit folgenden Vorgaben für Endpunkte in der Amazon Virtual Private Cloud für Amazon API Gateway erstellen:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Dies kann vom Inhalt Ihres Autorisierungs-Headers abhängen. Weitere Informationen finden Sie unter [the section called “Erwägungen zur VPC-Endpunktrichtlinie”](#). Weitere Beispielrichtlinien finden Sie unter [Beispiele für Datenperimeter-Richtlinien auf der Website](#). GitHub

Um die Richtlinie dem VPC-Endpunkt anzufügen, müssen Sie die VPC-Konsole verwenden. Weitere Informationen finden Sie unter [Steuern des Zugriffs auf Services mit VPC-Endpunkten](#).

Beispiel 1: VPC-Endpunktrichtlinie, die zwei Personen Zugriff gewährt APIs

Die folgende Beispielrichtlinie gewährt Zugriff auf nur zwei spezifische APIs über den VPC-Endpunkt, an den die Richtlinie angehängt ist.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "execute-api:Invoke"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*",
        "arn:aws:execute-api:us-east-1:123412341234:aaaaa11111/*"
      ]
    }
  ]
}
```

Beispiel 2: VPC-Endpunktrichtlinie, die Zugriff auf GET-Methoden gewährt

Die folgende Beispielrichtlinie gewährt Benutzern über den VPC-Endpunkt, dem die Richtlinie angefügt ist, Zugriff auf GET-Methoden für eine bestimmte API.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "execute-api:Invoke"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/
GET/*"
      ]
    }
  ]
}
```

```
]
}
```

Beispiel 3: VPC-Endpunktrichtlinie, die einem bestimmten Benutzer Zugriff auf eine bestimmte API gewährt

Die folgende Beispielrichtlinie gewährt über den VPC-Endpunkt, dem die Richtlinie angefügt ist, einem bestimmten Benutzer Zugriff auf eine bestimmte API.

In diesem Fall müssen Sie den `authorizationType` der Methode auf `AWS_IAM` oder `NONE` setzen, da die Richtlinie den Zugriff auf bestimmte IAM-Prinzipale einschränkt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::123412341234:user/MyUser"
        ]
      },
      "Action": [
        "execute-api:Invoke"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*"
      ]
    }
  ]
}
```

Beispiel 4: Eine VPC-Endpunktrichtlinie, die Benutzern Zugriff auf einen bestimmten benutzerdefinierten Domainnamen und alle der Domain zugeordneten API gewährt

Die folgende Beispielrichtlinie gewährt Benutzern Zugriff auf einen bestimmten benutzerdefinierten Domainnamen für private Zwecke APIs über den VPC-Endpunkt, an den die Richtlinie angehängt ist. Mit dieser Richtlinie kann der Benutzer, solange er eine Zuordnung für den Domainnamen-Zugriff

zwischen dem VPC-Endpunkt und dem benutzerdefinierten Domainnamen erstellt hat und Zugriff auf den benutzerdefinierten Domainnamen und alle privaten APIs erhält, die dem benutzerdefinierten Domainnamen zugeordnet sind, alle diesem benutzerdefinierten Domainnamen APIs zugewiesenen APIs aufrufen. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen für private APIs”](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnEquals": {
          "execute-api:viaDomainArn": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.test.com+f4g5h6",
        }
      }
    }
  ]
}
```

Beispiel 5: VPC-Endpunktrichtlinie, die den Zugriff auf bestimmte Ressourcen APIs und Domänenressourcen gewährt oder verweigert

Die folgende Beispielrichtlinie gewährt Benutzern Zugriff auf spezifische Ressourcen APIs und Domänenressourcen. Mit dieser Richtlinie kann der Benutzer zulässige private Ressourcen und Domänenressourcen aufrufen, solange er eine Zuordnung für den Domainnamen-Zugriff zwischen dem VPC-Endpunkt und dem benutzerdefinierten Domänennamen erstellt hat und Zugriff auf den benutzerdefinierten Domainnamen und alle privaten APIs erhält, die dem benutzerdefinierten

Domänennamen zugeordnet sind. APIs Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen für private APIs”](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.test.com+f4g5h6",
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/admin/*",
        "arn:aws:execute-api:us-west-2:111122223333:bcd123455/*"
      ]
    }
  ]
}
```

Beispiel 6: VPC-Endpunktrichtlinie, die Zugriff nach Prinzipalen und Ressourcen einer Organisation gewährt oder verweigert

Die folgende Beispiel-Richtlinie gewährt Zugriff auf Prinzipale und Ressourcen, die zu einer Organisation gehören.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringEquals": {
          "aws:ResourceOrgID": "o-abcd1234",
          "aws:PrincipalOrgID": "o-abcd1234"
        }
      },
      "Action": "*",
      "Resource": "*",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources"
    }
  ]
}
```

Verwenden Sie Tags, um den Zugriff auf REST APIs in API Gateway zu steuern

Die Zugriffsberechtigungen für REST APIs können mithilfe der attributbasierten Zugriffskontrolle in IAM-Richtlinien fein abgestimmt werden.

Weitere Informationen finden Sie unter [the section called “Attributbasierte Zugriffskontrolle”](#).

API Gateway-Lambda-Genehmiger verwenden

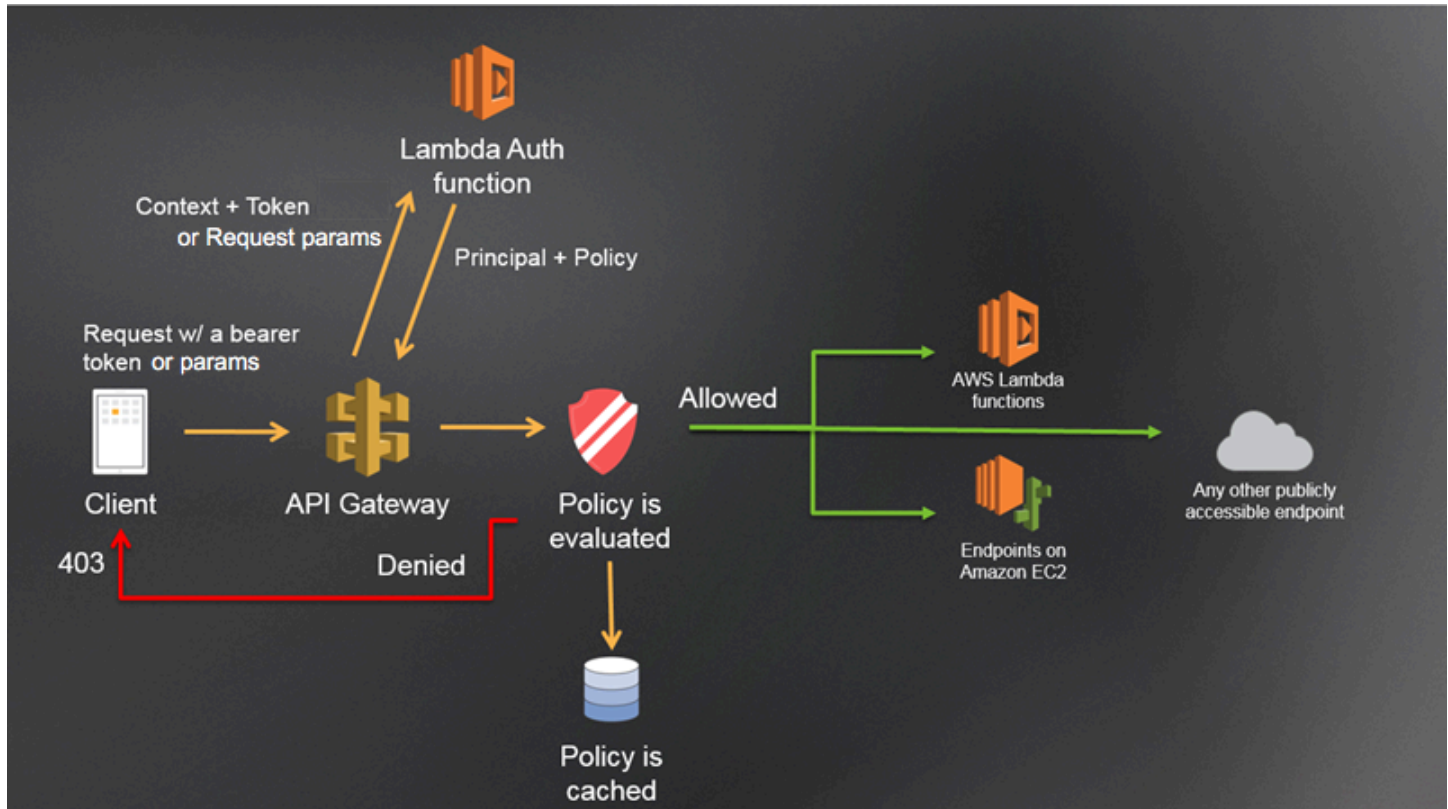
Verwenden Sie einen Lambda-Genehmiger (ehemals benutzerdefinierter Genehmiger) für die Zugriffskontrolle zu Ihrer API. Wenn ein Client eine Anfrage an die Methode Ihrer API stellt, ruft API Gateway Ihren Lambda-Genehmiger auf. Der Lambda-Genehmiger verwendet die Identität des Aufrufers als Eingabe und gibt eine IAM-Richtlinie als Ausgabe zurück.

Verwenden Sie einen Lambda-Genehmiger für die Implementierung eines benutzerdefinierten Genehmigungsschemas. Ihr Schema kann Anforderungsparameter verwenden, um die Identität des Anrufers zu ermitteln, oder eine Bearer-Token-Authentifizierungsstrategie wie SAML OAuth

verwenden. Erstellen Sie einen Lambda-Authorizer in der API Gateway REST API-Konsole, indem Sie das oder ein AWS CLI AWS SDK verwenden.

Genehmigungsworkflow für Lambda-Genehmiger

Im folgenden Diagramm sehen Sie den Genehmigungsworkflow für Lambda-Genehmiger.



API Gateway-Lambda-Autorisierungs-Workflow

1. Der Client ruft eine Methode für eine API-Gateway-API auf, wobei er ein Bearer-Token oder mehrere Anforderungsparameter übergibt.
2. API Gateway prüft, ob ein Lambda-Genehmiger für die Methodenanforderung konfiguriert ist. Ist dies der Fall, ruft API Gateway die Lambda-Funktion auf.
3. Die Lambda-Funktion authentifiziert den Aufrufer. Die Funktion kann sich wie folgt authentifizieren:
 - Indem Sie einen OAuth Anbieter anrufen, um ein OAuth Zugriffstoken zu erhalten.
 - Aufruf über einen SAML-Anbieter, um eine SAML-Zusicherung zu erhalten
 - Generierung einer IAM-Richtlinie basierend auf den Werten der Anforderungsparameter.
 - Abrufen von Anmeldeinformationen aus einer Datenbank

4. Die Lambda-Funktion gibt eine IAM-Richtlinie und eine Prinzipal-ID zurück. Wenn die Lambda-Funktion diese Informationen nicht zurückgibt, schlägt der Aufruf fehl.
5. API Gateway wertet die IAM-Richtlinie aus.
 - Wenn der Zugriff verweigert wird, gibt API Gateway einen geeigneten HTTP-Statuscode zurück, wie z. B. 403 ACCESS_DENIED.
 - Wenn der Zugriff gestattet wird, ruft API Gateway die Methode auf.

Wenn Sie Genehmigungs-Caching aktivieren, speichert API Gateway die Richtlinie zwischen, sodass die Lambda-Genehmigungsfunktion nicht erneut aufgerufen wird. Stellen Sie sicher, dass Ihre Richtlinie für alle Ressourcen und Methoden Ihrer API gilt.

Sie können die 403 ACCESS_DENIED oder die 401 UNAUTHORIZED-Gateway-Antworten anpassen. Weitere Informationen hierzu finden Sie unter [the section called “Gateway-Antworten”](#).

Auswahl eines Lambda-Genehmigertyps

Es gibt zwei Arten von Lambda-Genehmigern:

Parameterbasierten Lambda-Genehmiger anfordern (**REQUEST**-Genehmiger) anfordern

Ein REQUEST-Genehmiger empfängt die Identität des Aufrufers in einer Kombination aus Headern, Abfragezeichenfolgeparametern, [stageVariables](#) sowie [\\$context](#)-Variablen. Sie können einen REQUEST-Genehmiger zur Erstellung detaillierte Richtlinien auf Basis von Informationen aus verschiedenen Identitätsquellen wie den `$context.path`- und `$context.httpMethod`-Kontextvariablen verwenden.

Wenn Sie das Genehmigungs-Caching für einen REQUEST-Genehmiger aktivieren, überprüft API Gateway, ob alle angegebenen Identitätsquellen in der Anforderung vorhanden sind. Sollte eine angegebene Identifikationsquelle fehlen, Null oder leer sein, gibt API Gateway eine 401 Unauthorized-HTTP-Antwort zurück, ohne die Lambda-Genehmigerfunktion aufzurufen. Wenn mehrere Identitätsquellen definiert sind, werden sie alle für die Ableitung des Cache-Schlüssel des Genehmigers verwendet. Sie können einen detaillierten Cache-Schlüssel durch eine Verwendung mehrerer Identitätsquellen definieren.

Wenn Sie Teile des Cache-Schlüssels ändern und Ihre API erneut bereitstellen, verwirft der Genehmiger das im Cache befindliche Richtliniendokument und generiert ein neues Dokument.

Wenn Sie das Genehmigungs-Caching für einen REQUEST-Genehmiger deaktivieren, leitet API Gateway die Anforderung direkt an die Lambda-Funktion weiter.

Token-basierter Lambda-Genehmiger (**TOKEN**-Genehmiger)

Ein TOKEN Autorisierer erhält die Identität des Aufrufers in einem Trägertoken, z. B. einem JSON Web Token (JWT) oder einem Token. OAuth

Wenn Sie das Genehmigungs-Caching für einen TOKEN-Genehmiger aktivieren, wird der in der Token-Quelle angegebene Header-Name zum Cache-Schlüssel.

Darüber hinaus können Sie die Token-Validierung verwenden, um eine Anweisung einzugeben. RegEx API Gateway führt eine erste Überprüfung des Eingabe-Tokens für diesen Ausdruck durch und ruft nach erfolgreicher Validierung die Lambda-Genehmigerfunktion auf. Dies trägt dazu bei, die Anzahl der Aufrufe Ihrer API zu reduzieren.

Die `IdentityValidationExpression`-Eigenschaft wird ausschließlich für TOKEN-Genehmiger unterstützt. Weitere Informationen finden Sie unter [the section called “x-amazon-apigateway-authorizer”](#).

Note

Wir empfehlen, dass Sie einen REQUEST-Genehmiger für die Zugriffskontrolle auf Ihre API verwenden. Sie können den Zugriff auf Ihre API auf Basis mehrerer Identitätsquellen steuern, wenn Sie einen REQUEST-Genehmiger verwenden, anstatt einer einzigen Identitätsquelle, wenn Sie einen TOKEN-Genehmiger verwenden. Außerdem können Sie unterschiedliche Cache-Schlüssel mithilfe mehrerer Identitätsquellen für einen REQUEST-Genehmiger separieren.

Beispiel für eine Lambda-Funktion für **REQUEST**-Genehmiger

Der folgende Beispielcode erstellt eine Lambda-Funktion für Genehmiger, die eine Anforderung zulässt, wenn der vom Client bereitgestellte `HeaderAuth1-Header`, der `QueryString1-Abfrageparameter` und die Stufenvariable `StageVar1` jeweils den angegebenen Werten von `headerValue1`, `queryValue1` und `stageValue1` entsprechen.

Node.js

```
// A simple request-based authorizer example to demonstrate how to use request
```

```
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header, QueryString1
// query parameter, and stage variable of StageVar1 all match
// specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
// respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    } else {
        callback(null, generateDeny('me', event.methodArn));
    }
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
```

```

var authResponse = {};
authResponse.principalId = principalId;
if (effect && resource) {
    var policyDocument = {};
    policyDocument.Version = '2012-10-17'; // default version
    policyDocument.Statement = [];
    var statementOne = {};
    statementOne.Action = 'execute-api:Invoke'; // default action
    statementOne.Effect = effect;
    statementOne.Resource = resource;
    policyDocument.Statement[0] = statementOne;
    authResponse.policyDocument = policyDocument;
}
// Optional output with custom properties of the String, Number or Boolean type.
authResponse.context = {
    "stringKey": "stringval",
    "numberKey": 123,
    "booleanKey": true
};
return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}

```

Python

```

# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied headerauth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

def lambda_handler(event, context):
    print(event)

    # Retrieve request parameters from the Lambda function input:

```

```

headers = event['headers']
queryStringParameters = event['queryStringParameters']
pathParameters = event['pathParameters']
stageVariables = event['stageVariables']

# Parse the input for the parameter values
tmp = event['methodArn'].split(':')
apiGatewayArnTmp = tmp[5].split('/')
awsAccountId = tmp[4]
region = tmp[3]
restApiId = apiGatewayArnTmp[0]
stage = apiGatewayArnTmp[1]
method = apiGatewayArnTmp[2]
resource = '/'

if (apiGatewayArnTmp[3]):
    resource += apiGatewayArnTmp[3]

# Perform authorization to return the Allow policy for correct parameters
# and the 'Unauthorized' error, otherwise.

if (headers['HeaderAuth1'] == "headerValue1" and
queryStringParameters['QueryString1'] == "queryValue1" and
stageVariables['StageVar1'] == "stageValue1"):
    response = generateAllow('me', event['methodArn'])
    print('authorized')
    return response
else:
    print('unauthorized')
    response = generateDeny('me', event['methodArn'])
    return response
# Help function to generate IAM policy

def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect

```



```
statementOne['Resource'] = resource
policyDocument['Statement'] = [statementOne]
authResponse['policyDocument'] = policyDocument

authResponse['context'] = {
    "stringKey": "stringval",
    "numberKey": 123,
    "booleanKey": True
}

return authResponse

def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)

def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

In diesem Beispiel prüft die Lambda-Autorisierungsfunktion die Eingabeparameter und verhält sich wie folgt:

- Wenn alle erforderlichen Parameterwerte mit den erwarteten Werten übereinstimmen, gibt die Genehmiger-Funktion eine 200 OK-HTTP-Antwort und eine IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung wird erfolgreich ausgeführt:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Andernfalls gibt die Genehmigerfunktion eine 401 Unauthorized-HTTP-Antwort zurück, und die Methodenanforderung schlägt fehl.

Zusätzlich zur Rückgabe einer IAM-Richtlinie muss die Lambda-Funktion des Genehmigers auch die Prinzipal-ID des Aufrufers zurückgeben. Optional kann auch ein context-Objekt mit zusätzlichen Informationen zurückgegeben werden, die dem Integrations-Backend übergeben werden können. Weitere Informationen finden Sie unter [Ausgabe von einem API-Gateway-Lambda-Genehmiger](#).

Im Produktions-Code kann eine Authentifizierung des Benutzers erforderlich sein, bevor eine Autorisierung erfolgt. Sie können die Authentifizierungslogik zu der Lambda-Funktion hinzufügen, indem Sie einen Authentifizierungsanbieter aufrufen. Befolgen Sie hierzu die Anleitungen in der Dokumentation des Anbieters.

Beispiel für eine Lambda-Funktion für **TOKEN**-Genehmiger

Der folgende Beispielcode erstellt eine TOKEN Lambda-Genehmigungsfunktion, die es einem Aufrufer erlaubt, eine Methode aufzurufen, wenn der vom Client bereitgestellte Tokenwert allow lautet. Der Aufrufer darf die Anforderung nicht aufrufen, wenn der Tokenwert deny ist. Wenn der Tokenwert unauthorized lautet oder eine leere Zeichenfolge ist, gibt die Genehmigerfunktion eine 401 UNAUTHORIZED-Antwort zurück.

Node.js

```
// A simple token-based authorizer example to demonstrate how to use an
// authorization token
// to allow or deny a request. In this example, the caller named 'user' is allowed
// to invoke
// a request if the client-supplied token value is 'allow'. The caller is not
// allowed to invoke
// the request if the token value is 'deny'. If the token value is 'unauthorized' or
// an empty
// string, the authorizer function returns an HTTP 401 status code. For any other
// token value,
// the authorizer returns an HTTP 500 status code.
// Note that token values are case-sensitive.

export const handler = function(event, context, callback) {
    var token = event.authorizationToken;
    switch (token) {
        case 'allow':
            callback(null, generatePolicy('user', 'Allow', event.methodArn));
```

```

        break;
    case 'deny':
        callback(null, generatePolicy('user', 'Deny', event.methodArn));
        break;
    case 'unauthorized':
        callback("Unauthorized"); // Return a 401 Unauthorized response
        break;
    default:
        callback("Error: Invalid token"); // Return a 500 Invalid token response
    }
};

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    var authResponse = {};

    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17';
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke';
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }

    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

```

Python

```

# A simple token-based authorizer example to demonstrate how to use an authorization token

```

```
# to allow or deny a request. In this example, the caller named 'user' is allowed to
invoke
# a request if the client-supplied token value is 'allow'. The caller is not allowed
to invoke
# the request if the token value is 'deny'. If the token value is 'unauthorized' or
an empty
# string, the authorizer function returns an HTTP 401 status code. For any other
token value,
# the authorizer returns an HTTP 500 status code.
# Note that token values are case-sensitive.
```

```
import json
```

```
def lambda_handler(event, context):
    token = event['authorizationToken']
    if token == 'allow':
        print('authorized')
        response = generatePolicy('user', 'Allow', event['methodArn'])
    elif token == 'deny':
        print('unauthorized')
        response = generatePolicy('user', 'Deny', event['methodArn'])
    elif token == 'unauthorized':
        print('unauthorized')
        raise Exception('Unauthorized') # Return a 401 Unauthorized response
        return 'unauthorized'
    try:
        return json.loads(response)
    except BaseException:
        print('unauthorized')
        return 'unauthorized' # Return a 500 error
```

```
def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
```

```
policyDocument['Statement'] = [statementOne]
authResponse['policyDocument'] = policyDocument
authResponse['context'] = {
    "stringKey": "stringval",
    "numberKey": 123,
    "booleanKey": True
}
authResponse_JSON = json.dumps(authResponse)
return authResponse_JSON
```

Wenn in diesem Beispiel die API eine Methodenanfrage empfängt, übergibt API Gateway das Quell-Token an diese Lambda-Genehmigerfunktion im Attribut `event.authorizationToken`. Die Lambda-Genehmigerfunktion liest das Token und verhält sich wie folgt:

- Wenn der Token-Wert `allow` lautet, gibt die Genehmigerfunktion eine `200 OK`-HTTP-Antwort und eine IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung wird erfolgreich ausgeführt:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Wenn der Token-Wert `deny` lautet, gibt die Genehmigerfunktion eine `200 OK`-HTTP-Antwort und eine Deny-IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung schlägt fehl:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Action": "execute-api:Invoke",
  "Effect": "Deny",
  "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
}
```

 Note

Außerhalb der Testumgebung gibt API Gateway eine 403 Forbidden-HTTP-Antwort zurück und die Methodenanforderung schlägt fehl.

- Wenn der Token-Wert `unauthorized` lautet oder eine leere Zeichenfolge ist, gibt die Genehmigerfunktion eine 401 Unauthorized-HTTP-Antwort zurück, und der Methodenaufruf schlägt fehl.
- Wenn das Token anders lautet, erhält der Client eine 500 Invalid token-Antwort, und der Methodenaufruf schlägt fehl.

Zusätzlich zur Rückgabe einer IAM-Richtlinie muss die Lambda-Funktion des Genehmigers auch die Prinzipal-ID des Aufrufers zurückgeben. Optional kann auch ein `context`-Objekt mit zusätzlichen Informationen zurückgegeben werden, die dem Integrations-Backend übergeben werden können. Weitere Informationen finden Sie unter [Ausgabe von einem API-Gateway-Lambda-Genehmiger](#).

Im Produktions-Code kann eine Authentifizierung des Benutzers erforderlich sein, bevor eine Autorisierung erfolgt. Sie können die Authentifizierungslogik zu der Lambda-Funktion hinzufügen, indem Sie einen Authentifizierungsanbieter aufrufen. Befolgen Sie hierzu die Anleitungen in der Dokumentation des Anbieters.

Weitere Beispiele für Lambda-Genehmigerfunktionen

Die folgende Liste enthält weitere Beispiele für Lambda-Genehmigerfunktionen. Sie können eine Lambda-Funktion in demselben oder in einem anderen Konto als dem Konto erstellen, in dem Sie Ihre API erstellt haben.

Für die vorherigen Lambda-Beispielfunktionen können Sie die integrierten Funktionen verwenden [AWSLambdaBasicExecutionRole](#), da diese Funktionen keine anderen AWS Dienste aufrufen. Wenn Ihre Lambda-Funktion andere AWS Dienste aufruft, müssen Sie der Lambda-Funktion eine IAM-

Ausführungsrolle zuweisen. Folgen Sie beim Erstellen einer Rolle den Anweisungen unter [AWS Lambda -Ausführungsrolle](#).

Weitere Beispiele für Lambda-Genehmigerfunktionen

- Eine Beispielanwendung finden Sie unter [Open Banking Brasilien — Authorization Samples](#) on GitHub.
- Weitere Lambda-Beispielfunktionen finden Sie unter [aws-apigateway-lambda-authorizer-blueprints on](#). GitHub
- Sie können einen Lambda-Genehmiger erstellen, der Benutzer mithilfe von Amazon-Cognito-Benutzerpools authentifiziert und Aufrufer anhand eines RichtlinienSpeichers mithilfe Verified Permissions autorisiert. Weitere Informationen finden Sie unter [the section called “Zugriffssteuerung anhand der Attribute einer Identität mit Verified Permissions”](#).
- Die Lambda-Konsole stellt einen Python-Blueprint bereit, den Sie verwenden können, indem Sie Blueprint verwenden und dann Blueprint auswählen. `api-gateway-authorizer-python`

Einen API-Gateway-Lambda-Genehmiger konfigurieren

Nachdem Sie eine Lambda-Funktion erstellt haben, konfigurieren Sie die Lambda-Funktion als Genehmiger für Ihre API. Anschließend konfigurieren Sie Ihre Methode, sodass sie Ihren Lambda-Genehmiger aufruft, um festzustellen, ob ein Aufrufer Ihre Methode aufrufen kann. Sie können eine Lambda-Funktion in demselben oder in einem anderen Konto als dem Konto erstellen, in dem Sie Ihre API erstellt haben.

Sie können Ihren Lambda-Genehmiger mit in der API-Gateway-Konsole integrierten Tools oder mithilfe von [Postman](#) testen. Anleitungen zur Verwendung von Postman zum Testen Ihrer Lambda-Genehmigerfunktion finden Sie unter [the section called “Aufruf einer API mit Lambda-Genehmigern”](#).

Lambda-Genehmiger konfigurieren (Konsole)

Im folgenden Verfahren sehen Sie, wie Sie einen Lambda-Genehmiger in der API-Gateway-REST-API-Konsole erstellen. Weitere Informationen zu den verschiedenen Lambda-Genehmigertypen finden Sie unter [the section called “Auswahl eines Lambda-Genehmigertyps”](#).

REQUEST authorizer

So konfigurieren Sie einen **REQUEST**-Lambda-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API und dann Genehmiger aus.
3. Wählen Sie Genehmiger erstellen aus.
4. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
5. Wählen Sie als Genehmiger-Typ Lambda aus.
6. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Autorisierungsfunktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
7. Lassen Sie Lambda-Aufrufrolle leer, um der API-Gateway-REST-API-Konsole zu erlauben, eine ressourcenbasierte Richtlinie festzulegen. Die Richtlinie gewährt API Gateway Berechtigungen, die Lambda-Genehmigerfunktion aufzurufen. Alternativ können Sie den Namen einer IAM-Rolle eingeben, damit API Gateway die Lambda-Genehmigerfunktion aufrufen kann. Eine Beispielrolle finden Sie unter [Übernehmbare IAM-Rolle erstellen](#).
8. Wählen Sie für Lambda-Ereignisnutzlast die Option Anfrage.
9. Wählen Sie für Identitätsquellen-Typ einen Parametertyp aus. Die unterstützten Parametertypen sind Header, Query string, Stage variable und Context. Um weitere Identitätsquellen hinzuzufügen, wählen Sie Parameter hinzufügen.
10. Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn das Caching der Richtlinie aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert.

Wenn Sie Caching aktivieren, muss Ihr Genehmiger eine Richtlinie zurückgeben, die auf alle Methoden einer API anwendbar sind. Verwenden Sie die Kontextvariablen `$context.path` und `$context.httpMethod`, um eine methodenspezifische Richtlinie durchzusetzen.

11. Wählen Sie Autorisierer erstellen.

TOKEN authorizer

So konfigurieren Sie einen **TOKEN**-Lambda-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API und dann Genehmiger aus.
3. Wählen Sie Genehmiger erstellen aus.
4. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
5. Wählen Sie als Genehmiger-Typ Lambda aus.
6. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Autorisierungsfunktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
7. Lassen Sie Lambda-Aufrufrolle leer, um der API-Gateway-REST-API-Konsole zu erlauben, eine ressourcenbasierte Richtlinie festzulegen. Die Richtlinie gewährt API Gateway Berechtigungen, die Lambda-Genehmigerfunktion aufzurufen. Alternativ können Sie den Namen einer IAM-Rolle eingeben, damit API Gateway die Lambda-Genehmigerfunktion aufrufen kann. Eine Beispielrolle finden Sie unter [Übernehmbare IAM-Rolle erstellen](#).
8. Wählen Sie für Lambda-Ereignisnutzlast die Option Token aus.
9. Geben Sie unter Token-Quelle den Header-Namen ein, der das Genehmigungstoken enthält. Der Aufrufer muss einen Header mit diesem Namen enthalten, um das Genehmigungs-Token an den Lambda-Genehmiger zu senden.
10. (Optional) Geben Sie für die Token-Validierung eine Anweisung ein. RegEx API Gateway führt eine erste Überprüfung des eingegebenen Token für diesen Ausdruck an und ruft nach der erfolgreichen Validierung den Genehmiger auf.
11. Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn das Richtlinien-Caching aktiviert ist, wird der unter Token-Quelle angegebene Header-Name zum Cache-Schlüssel. Wenn das Caching der Richtlinie aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert.

Wenn Sie Caching aktivieren, muss Ihr Genehmiger eine Richtlinie zurückgeben, die auf alle Methoden einer API anwendbar sind. Sie können Autorisierer-Caching deaktivieren, wenn Sie eine methodenspezifische Richtlinie durchsetzen möchten.

12. Wählen Sie Autorisierer erstellen.

Nachdem Sie Ihren Lambda-Genehmiger erstellt haben, können Sie ihn testen. Im folgenden Verfahren sehen Sie, wie Sie Ihren Lambda-Genehmiger testen.

REQUEST authorizer

So testen Sie einen **REQUEST**-Lambda-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie den Namen Ihres Genehmigers aus.
3. Geben Sie unter Genehmiger testen einen Wert für Ihre Identitätsquelle ein.

Wenn Sie [the section called “Beispiel für eine Lambda-Funktion für REQUEST-Genehmiger”](#) verwenden, gehen Sie wie folgt vor:

- a. Wählen Sie Header, geben Sie **headerValue1** ein und wählen Sie dann Parameter hinzufügen aus.
- b. Wählen Sie unter Identitätsquellen-Typ die Option Abfragezeichenfolge, geben Sie **queryValue1** ein und wählen Sie dann Parameter hinzufügen aus.
- c. Wählen Sie unter Identitätsquellen-Typ die Option Stufenvariable aus und geben Sie **stageValue1** ein.

Sie können die Kontextvariablen für den Testaufruf nicht ändern, aber Sie können die API-Gateway-Genehmiger-Testereignisvorlage für Ihre Lambda-Funktion bearbeiten. Anschließend können Sie Ihre Lambda-Genehmigerfunktion mit geänderten Kontextvariablen testen. Weitere Informationen finden Sie unter [Lambda-Funktionen in der Konsole testen](#) im AWS Lambda -Entwicklerhandbuch.

4. Wählen Sie Genehmiger testen aus.

TOKEN authorizer

So testen Sie einen **TOKEN**-Lambda-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie den Namen Ihres Genehmigers aus.
3. Geben Sie unter Genehmiger testen einen Wert für Ihr Token ein.

Wenn Sie [the section called “Beispiel für eine Lambda-Funktion für TOKEN-Genehmiger”](#) verwenden, gehen Sie wie folgt vor:

- Geben Sie unter `authorizationToken` **allow** ein.

4. Wählen Sie Genehmiger testen aus.

Wenn Ihr Lambda-Genehmiger eine Anfrage in der Testumgebung erfolgreich abweist, wird eine `200 OK`-HTTP-Antwort zurückgegeben. Außerhalb der Testumgebung jedoch, gibt API Gateway eine `403 Forbidden`-HTTP-Antwort zurück und die Methodenanforderung schlägt fehl.

Lambda-Genehmiger konfigurieren (AWS CLI)

Der Befehl [create-authorizer](#) demonstriert den Vorgang der Erstellung eines Lambda-Genehmigers mithilfe der AWS CLI.

REQUEST authorizer

Der folgende [create-authorizer](#)-Befehl erstellt einen REQUEST-Genehmiger und verwendet den Authorizer-Header sowie die `accountId`-Kontextvariable als Identitätsquellen:

```
aws apigateway create-authorizer \
  --rest-api-id 1234123412 \
  --name 'First_Request_Custom_Authorizer' \
  --type REQUEST \
  --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
  --identity-source 'method.request.header.Authorization,context.accountId' \
  --authorizer-result-ttl-in-seconds 300
```

TOKEN authorizer

Der folgende [create-authorizer](#)-Befehl erstellt einen TOKEN-Genehmiger und verwendet den Authorization-Header als Identitätsquelle:

```
aws apigateway create-authorizer \
  --rest-api-id 1234123412 \
  --name 'First-Token-Custom-Authorizer' \
  --type TOKEN \
```

```
--authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
--identity-source 'method.request.header.Authorization' \
--authorizer-result-ttl-in-seconds 300
```

Nachdem Sie Ihren Lambda-Genehmiger erstellt haben, können Sie ihn testen. Der folgende [test-invoke-authorizer](#) Befehl testet einen Lambda-Authorizer:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \
--authorizer-id efg1234 \
--headers Authorization='Value'
```

Konfigurieren einer Methode für die Verwendung eines Lambda-Genehmigers (Konsole)

Nachdem Sie Ihren Lambda-Genehmiger konfiguriert haben, müssen Sie ihn einer Methode für Ihre API hinzufügen. Wenn Ihr Genehmiger Autorisierungs-Caching verwendet, stellen Sie sicher, dass Sie die Richtlinie aktualisieren, um den Zugriff auf die zusätzliche Methode zu kontrollieren.

So konfigurieren Sie eine API-Methode für die Verwendung eines Lambda-Genehmigers:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API aus.
3. Klicken Sie auf Ressourcen und wählen Sie dann eine neue oder bereits vorhandene Methode aus.
4. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
5. Wählen Sie für Genehmiger aus dem Drop-down-Menü den Lambda-Genehmiger aus, den Sie gerade erstellt haben.
6. (Optional) Wenn Sie das Genehmigungstoken an das Backend übergeben möchten, wählen Sie HTTP-Anforderungsheader aus. Wählen Sie Header hinzufügen und fügen Sie dann den Namen des Genehmigungsheaders hinzu. Geben Sie unter Name den Header-Namen ein, der mit dem Namen der Token-Quelle übereinstimmt, den Sie beim Erstellen des Lambda-Genehmigers für die API angegeben haben. Dieser Schritt gilt nicht für REQUEST-Genehmiger.
7. Wählen Sie Speichern.

8. Wählen Sie **Deploy API** aus, um die API für eine Stufe bereitzustellen. Für einen **REQUEST-Genehmiger** mit **Stufenvariablen** müssen Sie auch die erforderlichen **Stufenvariablen** definieren und ihre Werte auf der Seite **Stufen** angeben.

Konfigurieren einer Methode für die Verwendung eines Lambda-Genehmigers (AWS CLI)

Nachdem Sie Ihren Lambda-Genehmiger konfiguriert haben, müssen Sie ihn einer Methode für Ihre API hinzufügen. Sie können eine neue Methode erstellen oder einen Patch-Vorgang verwenden, um einen Genehmiger an eine vorhandene Methode anzuhängen. Wenn Ihr Genehmiger **Autorisierungs-Caching** verwendet, stellen Sie sicher, dass Sie die Richtlinie aktualisieren, um den Zugriff auf die zusätzliche Methode zu kontrollieren.

Der [put-method](#)-Befehl erstellt eine neue Methode, die einen Lambda-Genehmiger verwendet:

```
aws apigateway put-method --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --authorization-type CUSTOM \  
  --authorizer-id efg1234
```

Der [update-method](#)-Befehl aktualisiert eine vorhandene Methode, um einen Lambda-Genehmiger zu verwenden:

```
aws apigateway update-method \  
  --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --patch-operations op="replace",path="/authorizationType",value="CUSTOM"  
  op="replace",path="/authorizerId",value="efg1234"
```

Eingabe für einen API-Gateway-Lambda-Genehmiger

Im folgenden Abschnitt wird das Format der Eingabe aus API Gateway an einen Lambda-Genehmiger erklärt.

TOKEN-Eingabeformat

Für einen Lambda-Genehmiger (ehemals als **benutzerdefinierter Genehmiger** bezeichnet) des Typs **TOKEN** müssen Sie einen benutzerdefinierten Header als **Token Source** (Token-Quelle) angeben, wenn Sie den Genehmiger für Ihre API konfigurieren. Der API-Client muss das erforderliche

Autorisierungs-Token in diesem Header in der eingehenden Anforderung übergeben. Beim Empfang der eingehenden Methodenanfrage extrahiert API Gateway das Token aus dem benutzerdefinierten Header. Anschließend gibt es das Token als `authorizationToken`-Eigenschaft des `event`-Objekts der Lambda-Funktion zurück, zusätzlich zum Methode-ARN als `methodArn`-Eigenschaft:

```
{
  "type": "TOKEN",
  "authorizationToken": "{caller-supplied-token}",
  "methodArn": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}]/[{child-resources}]"
}
```

In diesem Beispiel gibt die `type`-Eigenschaft den Genehmigertyp an, wobei es sich um einen `TOKEN` Genehmiger handelt. Das `{caller-supplied-token}` stammt aus dem Autorisierungs-Header einer Client-Anforderung und kann eine beliebige Zeichenfolge sein. `methodArn` ist der ARN der eingehenden Methodenanfrage und wird von API Gateway entsprechend der Konfiguration des Lambda-Genehmigers aufgefüllt.

REQUEST-Eingabeformat

Für einen Lambda-Genehmiger vom Typ `REQUEST` übergibt API Gateway die Anfrageparameter als Teil des `event`-Objekts an die Lambda-Genehmigerfunktion. Die Anforderungsparameter umfassen Header, Pfad, Parameter für Abfragezeichenfolgen, Stufenvariablen und einige der Anforderungskontextvariablen. Der API-Aufrufer können die Pfadparameter, Header und Abfragezeichenfolgenparameter festlegen. Der API-Entwickler muss die Stage-Variablen während der API-Bereitstellung festlegen, und API Gateway stellt den Anfragekontext zur Laufzeit zur Verfügung.

Note

Pfadparameter können als Anfrageparameter an die Lambda-Genehmigerfunktion übergeben werden, aber sie können nicht als Identitätsquellen verwendet werden.

Das folgende Beispiel zeigt eine Eingabe für einen `REQUEST` Genehmiger für eine API-Methode (`GET /request`) mit einer Proxy-Integration:

```
{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
}
```

```

"resource": "/request",
"path": "/request",
"httpMethod": "GET",
"headers": {
  "X-AMZ-Date": "20170718T062915Z",
  "Accept": "*/*",
  "HeaderAuth1": "headerValue1",
  "CloudFront-Viewer-Country": "US",
  "CloudFront-Forwarded-Proto": "https",
  "CloudFront-Is-Tablet-Viewer": "false",
  "CloudFront-Is-Mobile-Viewer": "false",
  "User-Agent": "..."
},
"queryStringParameters": {
  "QueryString1": "queryValue1"
},
"pathParameters": {},
"stageVariables": {
  "StageVar1": "stageValue1"
},
"requestContext": {
  "path": "/request",
  "accountId": "123456789012",
  "resourceId": "05c7jb",
  "stage": "test",
  "requestId": "...",
  "identity": {
    "apiKey": "...",
    "sourceIp": "...",
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  }
},
"resourcePath": "/request",
"httpMethod": "GET",
"apiId": "abcdef123"
}

```

```
}
```

`requestContext` ist eine Zuordnung von Schlüssel-Wert-Paaren und entspricht der Variablen [\\$Kontext](#). Das Ergebnis hängt von der API ab.

API Gateway kann der Zuordnung u. U. neue Schlüssel hinzufügen. Weitere Informationen über die Eingabe von Lambda-Funktionen bei der Lambda-Proxy-Integration finden Sie unter [Eingabeformat einer Lambda-Funktion für die Proxy-Integration](#).

Ausgabe von einem API-Gateway-Lambda-Genehmiger

Die Ausgabe einer Lambda-Genehmigerfunktion ist ein wörterbuchähnliches Objekt, das die Prinzipal-ID (`principalId`) und ein Richtlinienendokument (`policyDocument`) mit einer Liste Richtlinienanweisungen umfassen muss. Die Ausgabe kann auch eine `context`-Zuweisung mit Schlüssel-Wert-Paaren umfassen. Wenn die API einen Nutzungsplan verwendet ([apiKeySource](#) ist auf `AUTHORIZER` gesetzt), muss die Lambda-Genehmigerfunktion einen der API-Schlüssel des Nutzungsplans als `usageIdentifierKey`-Eigenschaftswert ausgeben.

Hier ein Beispiel für diese Ausgabe.

JSON

```
{
  "principalId": "yyyyyyyy",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-
api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-
resources}]"
      }
    ]
  },
  "context": {
    "stringKey": "value",
    "numberKey": "1",
    "booleanKey": "true"
  },
  "usageIdentifierKey": "{api-key}"
}
```



```
}
```

Hier legt eine Richtlinienanweisung fest, ob dem API Gateway-Ausführungs-Service der Aufruf (Action) der angegebenen API-Methode (Resource) erlaubt oder verweigert wird (Effect). Je nach Ihrem Autorisierer müssen Sie möglicherweise den Zugriff auf mehrere Ressourcen kontrollieren. Sie können einen Platzhalter (*) verwenden, um einen Ressourcentyp (Methode) anzugeben. Weitere Informationen zum Festlegen von gültigen Richtlinien für das Aufrufen einer API finden Sie unter [Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway](#).

Für Methoden-ARNs mit Autorisierung, z. B. `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-resources}]` beträgt die maximale Länge 1 600 Bytes. Die Pfadparameterwerte, deren Größe zur Laufzeit ermittelt wird, können dazu führen, dass die Länge des ARN dieses Limit übersteigt. Wenn dies geschieht, empfängt der API-Client die Antwort 414 Request URI too long.

Darüber hinaus ist der Ressourcen-ARN, wie in der Richtlinienanweisungsausgabe des Genehmigers gezeigt, derzeit auf 512 Zeichen begrenzt. Aus diesem Grund können Sie die URI nicht mit einem längeren JWT-Token in einer Anforderungs-URI verwenden. Übergeben Sie das JWT-Token stattdessen sicher im Anforderungs-Header.

Sie können auf den `principalId`-Wert in einer Mapping-Vorlage unter Verwendung der `$context.authorizer.principalId`-Variablen zugreifen. Dies ist nützlich, wenn Sie den Wert an das Backend übergeben möchten. Weitere Informationen finden Sie unter [Kontextvariablen für Datentransformationen](#).

Sie können auf den `stringKey`-, `numberKey`- oder `booleanKey`-Wert (z. B. "value", "1" oder "true") der `context`-Zuweisung in einer Mapping-Vorlage zugreifen, indem Sie `$context.authorizer.stringKey`, `$context.authorizer.numberKey` bzw. `$context.authorizer.booleanKey` aufrufen. Die zurückgegebenen Werte werden alle in Text umgewandelt. Beachten Sie, dass Sie kein JSON-Objekt oder -Array als gültigen Wert eines Schlüssels in der `context`-Zuweisung festlegen können.

Sie können das `context`-Mapping verwenden, um im Cache abgelegte Anmeldeinformationen vom Genehmiger unter Verwendung einer Mapping-Vorlage für die Integrationsanfrage an das Backend zurückzugeben. Dies ermöglicht dem Backend, durch Nutzung der im Cache gespeicherten Anmeldeinformationen, um die Notwendigkeit für den Zugriff auf die geheimen Schlüssel zu reduzieren und die Autorisierungs-Token für jede Anforderung zu öffnen, eine verbesserte Benutzerumgebung bereitzustellen.

Für die Lambda-Proxy-Integration übergibt API Gateway das context-Objekt von einem Lambda-Genehmiger direkt an die Backend-Lambda-Funktion als Teil der Eingabe event. Sie können die context-Schlüssel-/Wert-Paare in der Lambda-Funktion durch den Aufruf von `$event.requestContext.authorizer.key` abrufen.

{api-key} steht für einen API-Schlüssel im Nutzungsplan der API-Stufe. Weitere Informationen finden Sie unter [the section called "Nutzungspläne"](#).

Im Folgenden wird die Beispielausgabe aus dem Beispiel-Lambda-Genehmiger gezeigt. Die Beispielausgabe enthält eine Richtlinienanweisung zum Blockieren (Deny) von GET Methodenaufrufen für die dev Phase einer API (ymy8tbxw7b) eines AWS Accounts ().
123456789012

JSON

```
{
  "principalId": "user",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/
GET/"
      }
    ]
  }
}
```

Aufruf einer API mit einem API Gateway-Lambda-Genehmiger

Nachdem Sie den Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bezeichnet) konfiguriert und die API bereitgestellt haben, sollten Sie die API mit aktiviertem Lambda-Genehmiger testen. Dazu benötigen Sie einen REST-Client, z. B. cURL oder [Postman](#). In den folgenden Beispielen verwenden wir Postman.

 Note

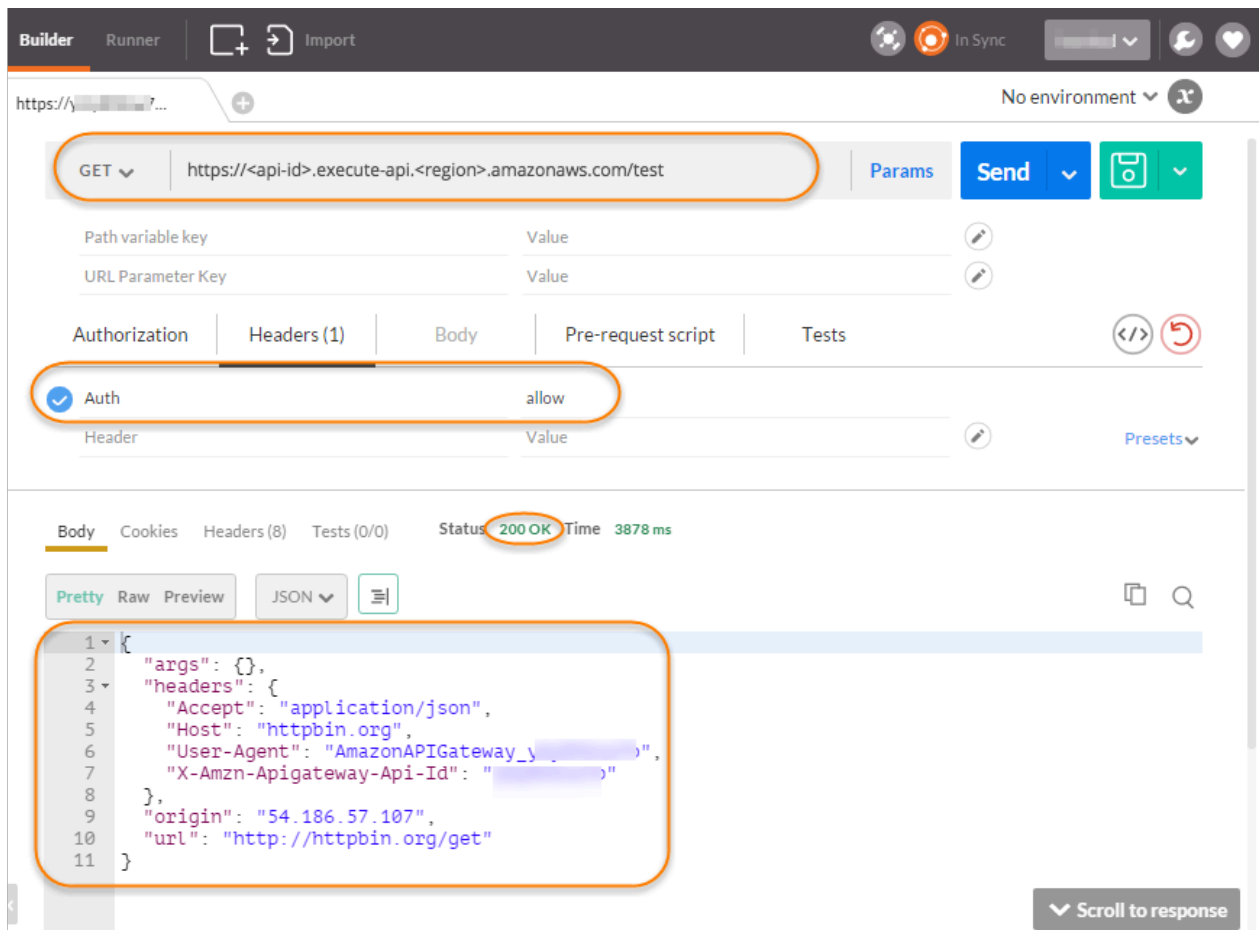
Beim Aufrufen einer autorisiererefähigen Methode protokolliert API Gateway den Aufruf nicht, CloudWatch wenn das erforderliche Token für den **TOKEN** Authorizer nicht gesetzt ist, Null ist oder durch den angegebenen Token-Validierungsausdruck ungültig gemacht wird. Ebenso protokolliert API Gateway den Aufruf nicht, CloudWatch wenn eine der erforderlichen Identitätsquellen für den REQUEST Autorisierer nicht festgelegt, Null oder leer ist.

Nachfolgend zeigen wir, wie Sie Postman verwenden, um mit einem Lambda-TOKEN-Genehmiger die API aufzurufen oder zu testen. Die Methode kann für das Aufrufen einer API mit einem Lambda-REQUEST-Genehmiger angewendet werden, wenn Sie den erforderlichen Parameter für Pfad, Header oder Abfragezeichenfolge explizit angeben.

Aufrufen einer API mit dem benutzerdefinierten **TOKEN** Genehmiger

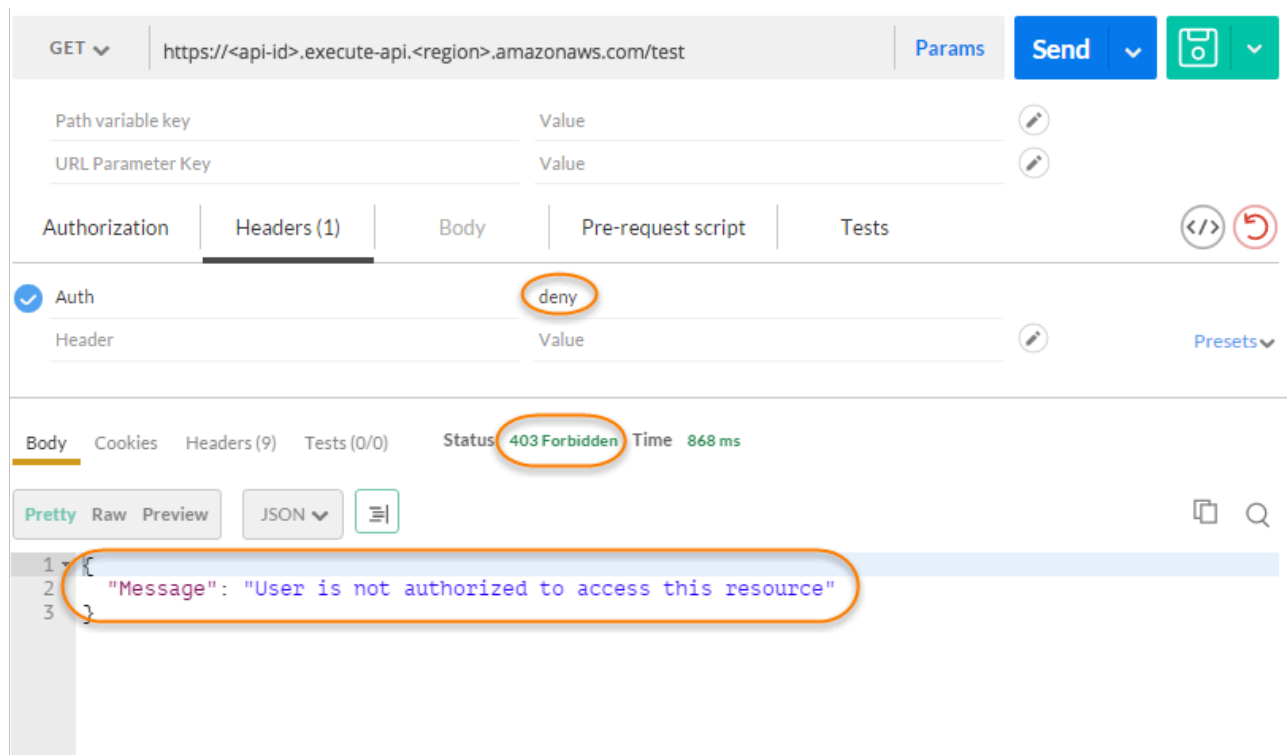
1. Öffnen Sie Postman, wählen Sie die GET-Methode aus und fügen Sie die Invoke URL der API im benachbarten URL-Feld ein.

Fügen Sie den Lambda-Genehmiger-Token-Header hinzu und setzen Sie den Wert auf `allow`. Wählen Sie Send (Senden) aus.



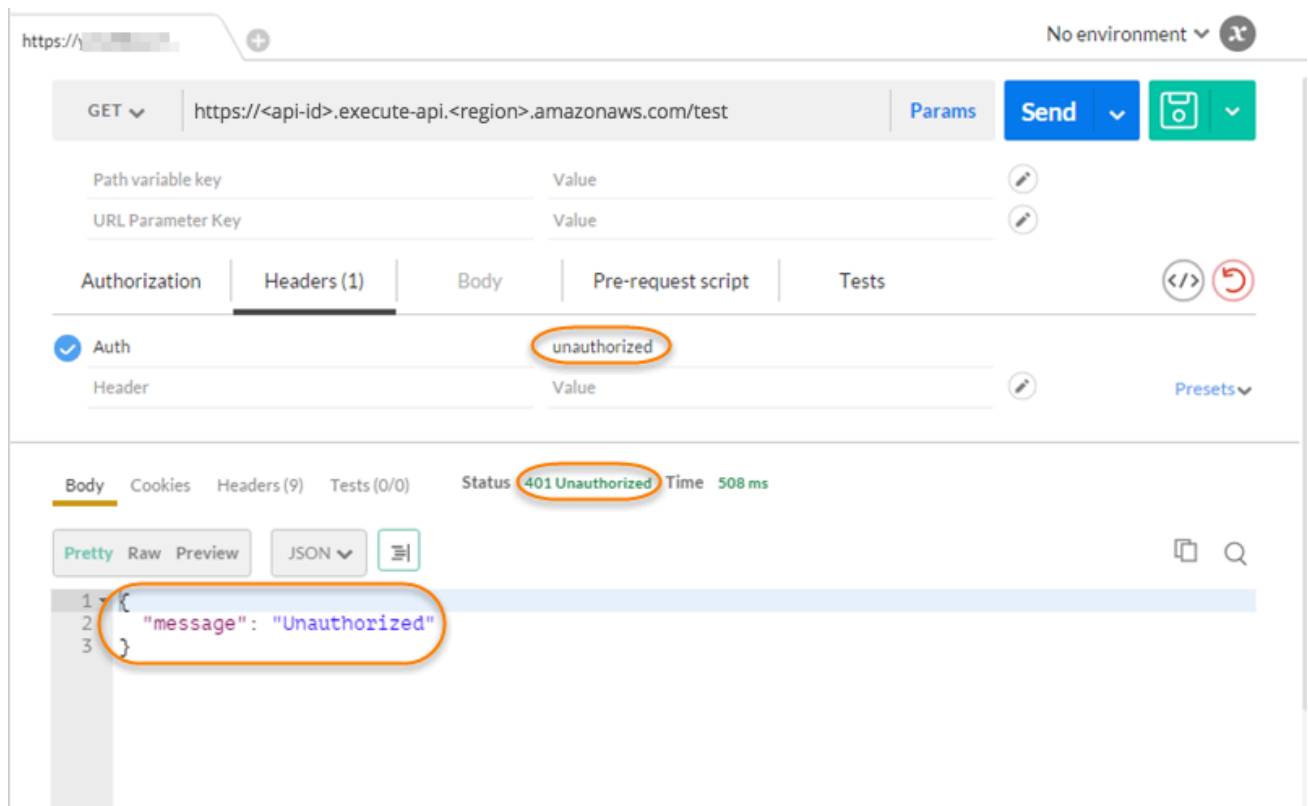
Die Antwort zeigt, dass der API Gateway-Lambda-Genehmiger die Antwort 200 OK zurückgibt und den Aufruf erfolgreich für den Zugriff auf den mit der Methode integrierten HTTP-Endpunkt (`http://httpbin.org/get`) autorisiert.

2. Ändern Sie in Postman den Lambda-Autorisierungstoken-Header-Wert in `deny`. Wählen Sie Send (Senden) aus.



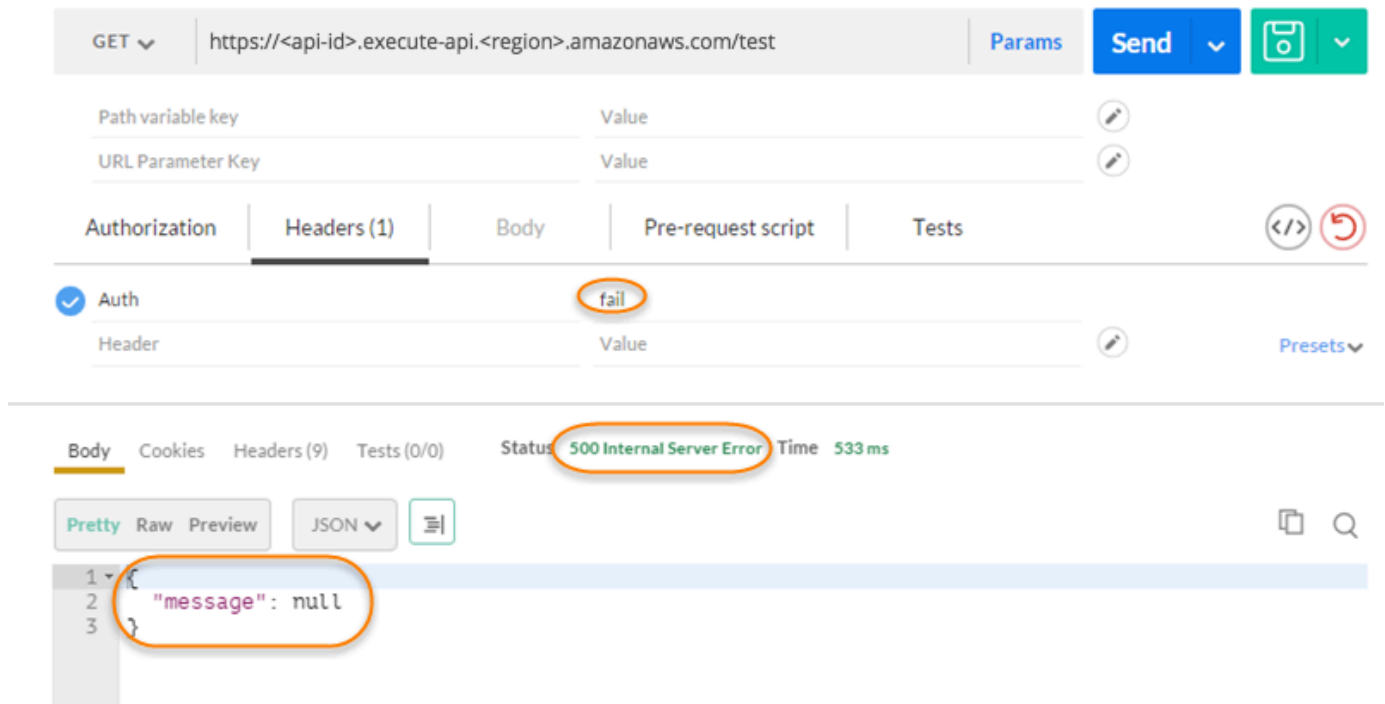
Die Antwort zeigt, dass der API Gateway-Lambda-Generierer die Antwort 403 Forbidden zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

3. Ändern Sie in Postman den Wert des Lambda-Autorisierungstoken-Headers in `unauthorized` und wählen Sie Send (Senden).



Die Antwort zeigt, dass API Gateway die Antwort 401 Unauthorized zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

4. Ändern Sie nun den Lambda-Genehmiger-Token-Header-Wert in `fail`. Wählen Sie Send (Senden) aus.



Die Antwort zeigt, dass API Gateway die Antwort 500 Internal Server Error zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

Konfigurieren eines kontenübergreifenden API Gateway-Lambda-Genehmigers

Sie können jetzt auch eine AWS Lambda Funktion aus einem anderen AWS Konto als API-Autorisierungsfunktion verwenden. Jedes Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist. Die Lambda-Autorisierungsfunktion kann Bearer-Token-Authentifizierungsstrategien wie OAuth SAML verwenden. Dies macht es einfach, eine zentrale Lambda-Autorisierungsfunktion zentral zu verwalten und über mehrere API Gateway gemeinsam zu nutzen. APIs

In diesem Abschnitt zeigen wir, wie eine kontoübergreifende Lambda-Genehmigerfunktion mit der Amazon API Gateway-Konsole konfiguriert wird.

Bei diesen Anweisungen wird davon ausgegangen, dass Sie bereits eine API-Gateway-API in einem AWS Konto und eine Lambda-Autorisierungsfunktion in einem anderen Konto haben.

Kontoübergreifenden Lambda-Genehmiger mit der API Gateway-Konsole konfigurieren

Melden Sie sich in der Amazon-API-Gateway-Konsole bei dem Konto an, in dem Ihre API enthalten ist, und gehen Sie wie folgt vor:

1. Wählen Sie Ihre API und dann im Hauptnavigationsbereich Genehmiger.
2. Wählen Sie Genehmiger erstellen aus.
3. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
4. Wählen Sie als Genehmiger-Typ Lambda aus.
5. Geben Sie für Lambda-Funktion den vollständigen ARN der Lambda-Genehmigerfunktion aus Ihrem zweiten Konto ein.

 Note

Sie finden den ARN für Ihre Funktion in der Lambda-Konsole in der oberen rechten Ecke des Konsolenfensters.

6. Eine Warnung mit einer `aws lambda add-permission`-Befehlszeichenfolge wird angezeigt. Diese Richtlinie gewährt API Gateway die Berechtigung, die Genehmiger-Lambda-Funktion aufzurufen. Kopieren Sie den Befehl und speichern Sie ihn für später. Führen Sie den Befehl aus, nachdem Sie den Genehmiger erstellt haben.
7. Wählen Sie für Lambda-Ereignisnutzlast entweder Token für einen TOKEN-Genehmiger oder Anfrage für einen REQUEST-Genehmiger.
8. Abhängig von der Auswahl im vorherigen Schritt führen Sie einen der folgenden Schritte aus:
 - a. Gehen Sie für die Option Token wie folgt vor:
 - Geben Sie unter Token-Quelle den Header-Namen ein, der das Genehmigungstoken enthält. Der API-Client muss einen Header mit diesem Namen enthalten, um das Autorisierungs-Token an den Lambda-Genehmiger zu senden.
 - Geben Sie optional für die Token-Validierung eine Anweisung ein RegEx . API Gateway führt eine erste Überprüfung des eingegebenen Token für diesen Ausdruck an und ruft nach der erfolgreichen Validierung den Genehmiger auf. Dies trägt dazu bei, die Anzahl der Aufrufe Ihrer API zu reduzieren.
 - Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn ein Caching der Richtlinie aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert. Wenn das Richtlinien-Caching aktiviert ist, wird der unter Token-Quelle angegebene Header-Name zum Cache-Schlüssel. Wenn in der Anfrage mehrere Werte an diesen Header übergeben werden, werden alle Werte zum Cache-Schlüssel, wobei die Reihenfolge beibehalten wird.

 Note

Die TTL-Standardwert ist 300 Sekunden. Der Höchstwert ist 3.600 Sekunden; dieses Limit kann nicht erhöht werden.

b. Für die Optionen Request machen Sie Folgendes:

- Wählen Sie für Identitätsquellen-Typ einen Parametertyp aus. Die unterstützten Parametertypen sind `Header`, `Query string`, `Stage variable` und `Context`. Um weitere Identitätsquellen hinzuzufügen, wählen Sie Parameter hinzufügen.
- Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn ein Caching der Richtlinie aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung `TTL = 0` wird das Richtlinien-Caching deaktiviert.

API Gateway verwendet die angegebenen Identitätsquellen als Cachingschlüssel für den Anfrageautorisierer. Wenn das Caching aktiviert ist, ruft API Gateway die Lambda-Funktion des Genehmigers erst auf, nachdem erfolgreich überprüft wurde, dass alle angegebenen Identitätsquellen zur Laufzeit vorhanden sind. Wenn eine angegebene Identitätsquelle fehlt, Null oder leer ist, gibt API Gateway eine `401 Unauthorized`-Antwort zurück, ohne die Lambda-Genehmigerfunktion aufzurufen.

Wenn mehrere Identitätsquellen definiert sind, werden sie alle verwendet, um den Cache-Schlüssel des Genehmigers abzuleiten. Das Ändern von Teilen des Cache-Schlüssels bewirkt, dass der Genehmiger das im Cache befindliche Richtliniendokument verwirft und ein neues erstellt. Wenn in der Anfrage ein Header mit mehreren Werten übergeben wird, werden alle Werte Teil des Cache-Schlüssels, wobei die Reihenfolge beibehalten wird.

- Wenn das Caching deaktiviert ist, ist es nicht erforderlich, eine Identitätsquelle anzugeben.

 Note

Zum Aktivieren der Zwischenspeicherung muss der Genehmiger eine Richtlinie zurückgeben, die für alle Methoden einer API anwendbar ist. Um methodenspezifische Richtlinien durchzusetzen, können Sie Genehmiger-Caching deaktivieren.

9. Wählen Sie Genehmiger erstellen aus.
10. Fügen Sie die `aws lambda add-permission` Befehlszeichenfolge, die Sie in einem vorherigen Schritt kopiert haben, in ein AWS CLI Fenster ein, das für Ihr zweites Konto konfiguriert ist. Ersetzen Sie `AUTHORIZER_ID` durch die ID Ihres Genehmigers. Dadurch erhält Ihr erstes Konto Zugriff auf die Lambda-Genehmigerfunktion Ihres zweiten Kontos.

Zugriffssteuerung anhand der Attribute einer Identität mit Verified Permissions

Mit Amazon Verified Permissions steuern Sie den Zugriff auf Ihre API-Gateway-API. Wenn Sie API Gateway mit Verified Permissions verwenden, wird ein Lambda-Genehmiger erstellt, der detaillierte Genehmigungsentscheidungen verwendet, um den Zugriff auf Ihre API zu steuern. Verified Permissions autorisiert Aufrufer nach einem Richtlinien Speicherschema, wobei die Cedar-Richtliniensprache verwendet wird, um detaillierte Berechtigungen für Anwendungsbenutzer zu definieren. Weitere Informationen finden Sie unter [Erstellen eines Richtlinien Speichers mit einer verbundenen API und einem Identitätsanbieter](#) im Benutzerhandbuch zu Amazon Verified Permissions.

Verified Permissions unterstützt Amazon-Cognito-Benutzerpools oder OpenID-Connect-(OIDC)-Identitätsanbieter als Identitätsquellen. Verified Permissions setzt voraus, dass der Principal bereits identifiziert und authentifiziert wurde. Verifizierte Berechtigungen werden nur für regionales und Edge-optimiertes REST unterstützt. APIs

Lambda-Genehmiger mit Verified Permissions erstellen

Verified Permissions erstellt einen Lambda-Genehmiger, um festzustellen, ob ein Principal eine Aktion an Ihrer API ausführen darf. Sie erstellen die Cedar-Richtlinie, die Verified Permissions zur Ausführung seiner Autorisierungsaufgaben verwendet.

Im Folgenden sehen Sie ein Beispiel für eine Cedar-Richtlinie, die den Zugriff auf den Aufruf einer API auf Basis des Amazon-Cognito-Benutzerpools `us-east-1_ABC1234` für die `developer`-Gruppe auf der `GET /users`-Ressource einer API erlaubt. Verified Permissions verifiziert die Gruppenmitgliedschaft, indem das Bearer-Token für die Identität des Anrufers analysiert wird.

```
permit(  
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",  
  action in [ MyAPI::Action::"get /users" ],  
  resource  
);
```

Optional kann Verified Permissions den Genehmiger an die Methoden Ihrer API anhängen. In Produktionsstufen Ihrer API empfehlen wir, Verified Permissions nicht zu erlauben, den Genehmiger für Sie anzuhängen.

In der folgenden Liste sehen Sie, wie Sie Verified Permissions so konfigurieren, dass der Lambda-Genehmiger an die Methodenanforderung der Methoden Ihrer API angehängt oder nicht angehängt wird.

Genehmiger für Sie anhängen (AWS-Managementkonsole)

Wenn Sie in der Verified-Permissions-Konsole die Option Richtlinienpeicher erstellen auswählen, klicken Sie anschließend auf der Seite App-Integration bereitstellen auf Jetzt.

Genehmiger nicht für Sie anhängen (AWS-Managementkonsole)

Wenn Sie in der Verified-Permissions-Konsole die Option Richtlinienpeicher erstellen auswählen, klicken Sie anschließend auf der Seite App-Integration bereitstellen auf Später.

Verified Permissions erstellt weiterhin einen Lambda-Genehmiger für Sie. Der Lambda-Genehmiger beginnt mit `AVPAuthorizerLambda-`. Weitere Anleitungen zum Anhängen Ihres Genehmigers an eine Methode finden Sie unter [the section called “Konfigurieren einer Methode für die Verwendung eines Lambda-Genehmigers \(Konsole\)”](#).

Genehmiger für Sie anhängen (CloudFormation)

Stellen Sie in der von Verified Permissions generierten CloudFormation Vorlage im Conditions Abschnitt auf ein. `"Ref": "shouldAttachAuthorizer" true`

Genehmiger nicht für Sie anhängen (CloudFormation)

Stellen Sie in der von Verified Permissions generierten CloudFormation Vorlage im Abschnitt den Wert auf ein. `Conditions "Ref": "shouldAttachAuthorizer" false`

Verified Permissions erstellt weiterhin einen Lambda-Genehmiger für Sie. Der Lambda-Genehmiger beginnt mit `AVPAuthorizerLambda-`. Weitere Anleitungen zum Anhängen Ihres Genehmigers an eine Methode finden Sie unter [the section called “Konfigurieren einer Methode für die Verwendung eines Lambda-Genehmigers \(AWS CLI\)”](#).

Lambda-Genehmiger mit Verified Permissions aufrufen

Sie können Ihren Lambda-Genehmiger aufrufen, indem Sie im Authorization-Header ein Identitäts- oder Zugriffstoken angeben. Weitere Informationen finden Sie unter [the section called “Aufruf einer API mit Lambda-Genehmigern”](#).

API Gateway speichert die Richtlinie, die Ihr Lambda-Genehmiger zurückgibt, 120 Sekunden lang im Cache. Sie können die TTL in der API-Gateway-Konsole oder mithilfe von AWS CLI ändern.

Steuern Sie den Zugriff auf REST APIs mithilfe von Amazon Cognito Cognito-Benutzerpools als Autorisierer

Als Alternative zur Verwendung von [IAM-Rollen und Richtlinien](#) oder eines [Lambda-Genehmigers](#) (früher als benutzerdefinierter Genehmiger bekannt), können Sie einen [Amazon Cognito-Benutzerpool](#) verwenden, um den Zugriff auf Ihre API in Amazon API Gateway zu steuern.

Um einen Amazon Cognito-Benutzerpool mit Ihrer API zu verwenden, müssen Sie zunächst einen Genehmiger vom Typ COGNITO_USER_POOLS erstellen und dann eine API-Methode zur Verwendung dieses Genehmigers konfigurieren. Nach der Bereitstellung der API muss sich der Client zunächst am Benutzerpool anmelden, ein [Identitäts- oder Zugriffstoken](#) für den Benutzer abrufen und dann die API-Methode mit einem dieser Token, die in der Regel auf den Authorization-Header der Anfrage festgelegt sind, aufrufen. Der API-Aufruf ist nur erfolgreich, wenn der erforderliche Token übergeben und gültig ist. Andernfalls wird der Client nicht für den Aufruf autorisiert, da der Client über keine autorisierbaren Anmeldeinformationen verfügt.

Mit dem Identitätstoken werden API-Aufrufe auf Grundlage der Identitätsansprüche des angemeldeten Benutzers autorisiert. Mit dem Zugriffstoken werden API-Aufrufe auf Grundlage der benutzerdefinierten Bereiche von angegebenen zugriffsgeschützten Ressourcen autorisiert. Weitere Informationen finden Sie unter [Verwenden von Token mit Benutzerpools](#) und [Ressourcenserver und benutzerdefinierte Bereiche](#).

Um einen Amazon Cognito-Benutzerpool für Ihre API zu erstellen und zu konfigurieren, führen Sie die folgenden Aufgaben aus:

- Verwenden Sie die Amazon Cognito Cognito-Konsole, das CLI/SDK oder die API, um einen Benutzerpool zu erstellen — oder verwenden Sie einen Pool, der einem anderen Konto gehört. **AWS**
- Verwenden Sie die API Gateway-Konsole, die CLI, das SDK oder die API, um einen API Gateway-Genehmiger mit dem ausgewählten Benutzerpool zu erstellen.
- Verwenden Sie die API Gateway-Konsole, die CLI, das SDK oder die API, um den Genehmiger für ausgewählte API-Methoden zu aktivieren.

Um alle API-Methoden mit einem Benutzerpool zu aktivieren, führen Ihre API-Clients die folgenden Aufgaben aus:

- Verwenden Sie Amazon Cognito CLI/SDK oder API, um einen Benutzer im ausgewählten Benutzerpool anzumelden und ein Identitäts- oder Zugriffstoken zu erhalten. Weitere Informationen zur Verwendung von finden Sie unter [Codebeispiele für die SDKs Verwendung AWS SDKs von Amazon Cognito](#).
- Verwenden Sie ein kundenspezifisches Framework, um die bereitgestellte API Gateway-API aufzurufen und das entsprechende Token im Authorization-Header bereitzustellen.

Als API-Entwickler müssen Sie Ihren Client-Entwicklern die Benutzerpool-ID, eine Client-ID und die entsprechenden Clientschlüssel bereitstellen, die als Teil des Benutzerpools definiert werden.

Note

Verwenden Sie [Amazon Cognito Federated Identities](#), damit sich ein Benutzer mit Anmeldeinformationen von Amazon Cognito anmelden und temporäre Anmeldeinformationen zur Verwendung mit den Berechtigungen einer IAM-Rolle erhalten kann. Legen Sie bei jeder HTTP-Methode für API-Ressourcenendpunkte den Autorisierungstyp, Kategorie Method Execution auf AWS_IAM fest.

In diesem Abschnitt wird beschrieben, wie Sie einen Benutzerpool erstellen, eine API Gateway-API in den Benutzerpool integrieren und eine API aufrufen, die im Benutzerpool integriert ist.

Themen

- [Amazon Cognito-Benutzerpool für eine REST-API erstellen](#)
- [REST-API mit einem Amazon Cognito-Benutzerpool integrieren](#)
- [Mit einem Amazon Cognito-Benutzerpool integrierte REST-API aufrufen](#)
- [Kontoübergreifenden Amazon Cognito-Genehmiger für eine REST-API über die API Gateway-Konsole konfigurieren](#)
- [Erstellen Sie einen Amazon Cognito Cognito-Authorizer für eine REST-API mit CloudFormation](#)

Amazon Cognito-Benutzerpool für eine REST-API erstellen

Bevor Sie Ihre API in einen Benutzerpool integrieren, müssen Sie den Benutzerpool in Amazon Cognito erstellen. Ihre Benutzerpoolkonfiguration muss alle [Ressourcenkontingente für Amazon Cognito](#) einhalten. Alle benutzerdefinierten Amazon-Cognito-Variablen wie Gruppen, Benutzer und Rollen dürfen nur alphanumerische Zeichen verwenden. Eine Anleitung zum Erstellen eines

Benutzerpools finden Sie unter [Tutorial: Erstellen eines Benutzerpools](#) im Amazon-Cognito-Entwicklerhandbuch.

Beachten Sie die Benutzerpool-ID, Client-ID und jeden Clientschlüssel. Der Client muss diese an Amazon Cognito übermitteln, um Benutzer für den Benutzerpool zu registrieren, die Anmeldung am Benutzerpool auszuführen und der Identitäts- oder Zugriffstoken abzurufen, das in die Anfragen für Aufrufe von API-Methoden eingebunden werden muss, die für den Benutzerpool konfiguriert sind. Außerdem müssen Sie den Namen des Benutzerpools angeben, wenn Sie den Benutzerpool als Genehmiger in API Gateway konfigurieren, wie nachfolgend beschrieben.

Wenn Sie API-Methodenaufrufe mit Zugriffstoken autorisieren, stellen Sie bei der Konfiguration der App-Integration beim Benutzerpool sicher, dass die für einen bestimmten Ressourcenserver gewünschten benutzerdefinierten Bereiche eingerichtet werden. Weitere Informationen über die Verwendung von Tokens mit Amazon Cognito-Benutzerpools finden Sie unter [Tokens mit Benutzerpools verwenden](#). Weitere Informationen zu Ressourcenservern finden Sie unter [Definieren von Ressourcenservern für eigene Benutzerpools](#).

Beachten Sie die konfigurierten Ressourcenserver-IDs und benutzerdefinierten Bereichsnamen. Sie benötigen sie, um die vollständigen Namen des Zugriffsbereichs für OAuth Bereiche zu erstellen, die COGNITO_USER_POOLS vom Autorisierer verwendet werden.

The screenshot shows the Amazon Cognito console interface. The breadcrumb navigation at the top reads: **Amazon Cognito** > **User pools** > **UserPool6BA7E5F2-H9t6LPh3O3XL** > **Domain**. On the left sidebar, the 'Branding' section is expanded, with 'Domain' selected. The main content area is titled 'Domain' and contains two panels: 'Cognito domain' and 'Custom domain'. Below these is the 'Resource servers (1)' section, which includes a search bar and a table of configured resource servers. One resource server, 'PetStore', is listed with its identifier 'https://my-petstore-api.example.com' and '2 custom scopes'. Red boxes highlight these two fields in the table.

Domain

Domain Info Actions

Configure a domain for managed login. A domain is required for authentication with managed login pages, third-party IdP authentication, and operations where your user pool functions as an OIDC IdP.

Cognito domain	Custom domain
Domain -	Domain -
Branding version -	Branding version -

Resource servers (1) Info Edit Delete Create resource server

Configure resource servers. A resource server is a remote server that authorizes access based on OAuth 2.0 scopes in an access token.

Search resource servers by name or ID

Resource server name	Resource server identifier	Custom scopes
☐ PetStore	https://my-petstore-api.example.com	2 custom scopes

REST-API mit einem Amazon Cognito-Benutzerpool integrieren

Nach der Erstellung eines Amazon Cognito-Benutzerpools müssen Sie in API Gateway einen `COGNITO_USER_POOLS`-Genehmiger erstellen, der den Benutzerpool verwendet. Im folgenden Verfahren wird gezeigt, wie Sie dies mithilfe der API-Gateway-Konsole tun.

Note

Sie können mit der [CreateAuthorizer](#)-Aktion einen `COGNITO_USER_POOLS`-Autorisierer erstellen, der mehrere Benutzerpools verwendet. Sie können bis zu 1 000 Benutzerpools für einen `COGNITO_USER_POOLS`-Autorisierer verwenden. Dieses Limit kann nicht erhöht werden.

Important

Nachdem Sie eines der folgenden Verfahren durchgeführt haben, müssen Sie Ihre API bereitstellen oder erneut bereitstellen, um die Änderungen zu übernehmen. Weitere Informationen zum Bereitstellen Ihrer API finden Sie unter [REST-APIs in API Gateway bereitstellen](#).

So erstellen Sie einen **COGNITO_USER_POOLS**-Genehmiger unter Verwendung der API Gateway-Konsole:

1. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API in API Gateway aus.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Wählen Sie Genehmiger erstellen aus.
4. Zur Konfiguration des neuen Genehmigers für die Verwendung eines Benutzerpools führen Sie die folgenden Schritte aus:
 - a. Geben Sie unter Name des Genehmigers einen Namen ein.
 - b. Wählen Sie als Genehmiger-Typ Cognito aus.
 - c. Wählen Sie für den Cognito-Benutzerpool den Ort aus, AWS-Region an dem Sie Ihren Amazon Cognito erstellt haben, und wählen Sie einen verfügbaren Benutzerpool aus.

Mit einer Stufenvariablen können Sie Ihren Benutzerpool definieren. Verwenden Sie das folgende Format für den Benutzerpool: `arn:aws:cognito-idp:us-east-2:111122223333:userpool/${stageVariables.MyUserPool}`.

- d. Geben Sie für Token-Quelle als Header-Name **Authorization** ein, um das Identitäts- oder Zugriffstoken zu übergeben, das von Amazon Cognito bei erfolgreicher Anmeldung eines Benutzers zurückgegeben wird.
 - e. Optional können Sie einen regulären Ausdruck im Feld Tokenvalidierung eingeben, um das aud-Feld (Zielgruppe) des Identitätstokens auszuwerten, bevor die Anfrage mit Amazon Cognito genehmigt wird. Beachten Sie, dass diese Validierung bei Verwendung eines Zugriffstoken die Anforderung zurückweist, da das Zugriffstoken das aud-Feld nicht enthält.
 - f. Wählen Sie Genehmiger erstellen aus.
5. Nach der Erstellung des COGNITO_USER_POOLS-Genehmigers können Sie einen Testaufruf durchführen, indem Sie ein vom Benutzerpool bereitgestelltes Identitätstoken übergeben. Sie können kein Zugriffstoken verwenden, um Ihren Autorisierer testweise aufzurufen.

Sie können dieses Identitätstoken erhalten, indem Sie das [Amazon Cognito Identity-SDK](#) aufrufen, um die Benutzeranmeldung durchzuführen. Sie können auch die [InitiateAuth](#)-Aktion verwenden. Wenn Sie keine Genehmigungsbereiche konfigurieren, behandelt API Gateway das bereitgestellte Token als Identitätstoken.

Das vorhergehende Verfahren erstellt einen COGNITO_USER_POOLS-Genehmiger, der den neu erstellten Amazon Cognito-Benutzerpool verwendet. Je nach der Aktivierungsweise des Genehmigers mit einer API-Methode können Sie entweder ein Identitäts- oder Zugriffstoken verwenden, das vom integrierten Benutzerpool bereitgestellt wird.

So konfigurieren Sie einen **COGNITO_USER_POOLS**-Genehmiger mit Methoden

1. Wählen Sie Resources aus. Wählen Sie eine neue oder eine vorhandene Methode aus. Falls erforderlich, erstellen Sie eine Ressource.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Wählen Sie für Genehmiger aus dem Dropdown-Menü die Amazon-Cognito-Benutzerpool-Genehmiger, die Sie soeben erstellt haben.
4. Zur Verwendung eines Identitätstokens führen Sie die folgenden Schritte aus:

- a. Lassen Sie das Feld Genehmigungsbereiche leer.
- b. Fügen Sie bei Bedarf unter Integrationsanforderung den Ausdruck `$context.authorizer.claims['property-name']` oder `$context.authorizer.claims.property-name` in einer Text-Mapping-Vorlage hinzu, um die angegebene Eigenschaft für Identitätsansprüche vom Benutzerpool an das Backend zu übergeben. Bei einfachen Eigenschaftsnamen (wie z. B. `sub` oder `custom-sub`) sind beide Notationen gleich. Bei komplexen Eigenschaftsnamen (wie z. B. `custom:role`) darf keine punktierte Schreibweise verwendet werden. Beispielsweise werden die [Standardfelder](#) `sub` und `email` des Anspruchs mit den folgenden Mapping-Ausdrücken an das Backend übergeben:

```
{
  "context" : {
    "sub" : "$context.authorizer.claims.sub",
    "email" : "$context.authorizer.claims.email"
  }
}
```

Wenn Sie bei der Konfiguration des Benutzerpools ein Feld für einen benutzerdefinierten Anspruch deklariert haben, können Sie auf die gleiche Art auf die benutzerdefinierten Felder zugreifen. Im folgenden Beispiel wird das benutzerdefinierte Feld `role` eines Anspruchs abgerufen:

```
{
  "context" : {
    "role" : "$context.authorizer.claims.role"
  }
}
```

Wurde das Feld für den benutzerdefinierten Anspruch mit `custom:role` deklariert, verwenden Sie folgendes Beispiel für den Abruf der Anspruchseigenschaft:

```
{
  "context" : {
    "role" : "$context.authorizer.claims['custom:role']"
  }
}
```

5. Zur Verwendung eines Zugriffstokens führen Sie die folgenden Schritte aus:
 - a. Geben Sie für Genehmigungsbereiche einen oder mehrere vollständige Namen eines Bereichs ein, der bei der Erstellung des Amazon-Cognito-Benutzerpools konfiguriert wurde. Im Beispiel, das in [Amazon Cognito-Benutzerpool für eine REST-API erstellen](#) gezeigt wurde, ist einer der Bereiche beispielsweise `https://my-petstore-api.example.com/cats.read`.

Bei Laufzeit ist der Methodenaufruf dann erfolgreich, wenn ein Bereich, der in diesem Schritt in der Methode angegeben wird, einem Bereich entspricht, der im eingehenden Token beantragt wird. Andernfalls schlägt der Aufruf mit einer 401 `Unauthorized`-Antwort fehl.

- b. Wählen Sie Speichern.
6. Wiederholen Sie diese Schritte für weitere Methoden Ihrer Wahl.

Wenn beim `COGNITO_USER_POOLS` Authorizer die Option `OAuthScopes` nicht angegeben ist, behandelt API Gateway das bereitgestellte Token als Identitätstoken und verifiziert die beanspruchte Identität anhand der Identität aus dem Benutzerpool. Andernfalls behandelt API Gateway den angegebenen Token als Zugriffstoken und prüft die im Token beanspruchten Zugriffsbereiche gegen die in der Methode deklarierten Autorisierungsbereiche.

Statt die API Gateway-Konsole zu verwenden, können Sie auch einen Amazon Cognito-Benutzerpool für eine Methode aktivieren, indem Sie eine OpenAPI-Definitionsdatei angeben und die API-Definition in API Gateway importieren.

So importieren Sie einen `COGNITO_USER_POOLS`-Genehmiger mit einer Swagger-Definitionsdatei

1. Erstellen (oder exportieren) Sie eine Swagger-Definitionsdatei für Ihre API.
2. Geben Sie die `COGNITO_USER_POOLS` -Genehmiger (`MyUserPool`) -Definition als Teil der JSON- `securitySchemes` Abschnitt in OpenAPI 3.0 oder den `securityDefinitions` Abschnitt im Open API-2.0 wie folgt an:

OpenAPI 3.0

```
"securitySchemes": {
  "MyUserPool": {
    "type": "apiKey",
    "name": "Authorization",
    "in": "header",
    "x-amazon-apigateway-authtype": "cognito_user_pools",
```

```

"x-amazon-apigateway-authorizer": {
  "type": "cognito_user_pools",
  "providerARNs": [
    "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
  ]
}
}

```

OpenAPI 2.0

```

"securityDefinitions": {
  "MyUserPool": {
    "type": "apiKey",
    "name": "Authorization",
    "in": "header",
    "x-amazon-apigateway-authtype": "cognito_user_pools",
    "x-amazon-apigateway-authorizer": {
      "type": "cognito_user_pools",
      "providerARNs": [
        "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
      ]
    }
  }
}

```

3. Wenn Sie den Identitätstoken für die Methodenautorisierung verwenden möchten, fügen Sie der { "MyUserPool": [] }-Definition der Methode security hinzu, wie in der folgenden GET-Methode auf der Root-Ressource dargestellt.

```

"paths": {
  "/": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "text/html"
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}

```

```

        }
      }
    },
    "security": [
      {
        "MyUserPool": []
      }
    ],
    "x-amazon-apigateway-integration": {
      "type": "mock",
      "responses": {
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.Content-Type": "'text/html'"
          },
        },
      },
    },
    "requestTemplates": {
      "application/json": "{$\"statusCode\": 200}"
    },
    "passthroughBehavior": "when_no_match"
  }
},
...
}

```

4. Um den Zugriffstoken für die Methodenautorisierung zu verwenden, ändern Sie die obige Sicherheitsdefinition in { "MyUserPool": [resource-server/scope, ...] }:

```

"paths": {
  "/": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "text/html"
      ],
      "responses": {
        "200": {
          "description": "200 response",

```

```

        "headers": {
            "Content-Type": {
                "type": "string"
            }
        }
    },
    "security": [
        {
            "MyUserPool": ["https://my-petstore-api.example.com/cats.read",
"http://my.resource.com/file.read"]
        }
    ],
    "x-amazon-apigateway-integration": {
        "type": "mock",
        "responses": {
            "default": {
                "statusCode": "200",
                "responseParameters": {
                    "method.response.header.Content-Type": "'text/html'"
                },
            }
        },
        "requestTemplates": {
            "application/json": "{$\"statusCode\": 200}"
        },
        "passthroughBehavior": "when_no_match"
    }
},
...
}

```

- Bei Bedarf können Sie mithilfe der geeigneten Swagger-Definitionen oder -Erweiterungen weitere API-Konfigurationseinstellungen festlegen. Weitere Informationen finden Sie unter [OpenAPI-Erweiterungen für API Gateway](#).

Mit einem Amazon Cognito-Benutzerpool integrierte REST-API aufrufen

Vom Client müssen die folgenden Aufgaben ausgeführt werden, um eine Methode mit konfigurierbarem Benutzerpool-Genehmiger aufzurufen:

- Der Benutzer muss für den Benutzerpool registriert sein.

- Der Benutzer muss sich am Benutzerpool anmelden.
- Rufen Sie ein [Identitäts- oder Zugriffs-Token](#) des angemeldeten Benutzers vom Benutzerpool ab.
- Das Token muss in den Authorization-Header (oder einen anderen Header, der beim Erstellen des Genehmigers festgelegt wurde) eingebunden werden.

Sie können [AWS Amplify](#) verwenden, um diese Aufgaben auszuführen. Weitere Informationen finden Sie unter [Integration von Amazon Cognito mit Web- und Mobile-Apps](#).

- Informationen zu Android finden Sie unter [Erste Schritte mit Amplify für Android](#).
- Informationen zur Verwendung von iOS finden Sie unter [Erste Schritte mit Amplify für iOS](#).
- Informationen zur Verwendung JavaScript finden Sie unter [Erste Schritte mit Amplify für Javascript](#).

Kontoübergreifenden Amazon Cognito-Genehmiger für eine REST-API über die API Gateway-Konsole konfigurieren

Sie können jetzt auch einen Amazon Cognito Cognito-Benutzerpool von einem anderen AWS Konto aus als API-Autorisierer verwenden. Der Amazon Cognito Cognito-Benutzerpool kann Bearer-Token-Authentifizierungsstrategien wie SAML OAuth verwenden. Dies macht es einfach, einen zentralen Amazon Cognito Cognito-Benutzerpool-AuthORIZER über mehrere API Gateway zentral zu verwalten und gemeinsam zu nutzen. APIs

In diesem Abschnitt zeigen wir, wie ein kontenübergreifender Amazon Cognito-Benutzerpool mit Hilfe der Amazon API Gateway-Konsole konfiguriert wird.

Bei diesen Anweisungen wird davon ausgegangen, dass Sie bereits eine API Gateway Gateway-API in einem AWS Konto und einen Amazon Cognito Cognito-Benutzerpool in einem anderen Konto haben.

Erstellen eines kontenübergreifenden Amazon-Cognito-Genehmigers für eine REST-API

Melden Sie sich in der Amazon-API-Gateway-Konsole bei dem Konto an, in dem Ihre API enthalten ist, und gehen Sie wie folgt vor:

1. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API in API Gateway aus.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Wählen Sie Genehmiger erstellen aus.

4. Zur Konfiguration des neuen Genehmigers für die Verwendung eines Benutzerpools führen Sie die folgenden Schritte aus:
 - a. Geben Sie unter Name des Genehmigers einen Namen ein.
 - b. Wählen Sie als Genehmiger-Typ Cognito aus.
 - c. Geben Sie den vollständigen ARN für den Benutzerpool in Ihrem zweiten Konto in das Feld Cognito-Benutzerpool ein.

 Note

In der Amazon Cognito-Konsole finden Sie den ARN für eigene Benutzerpools im Feld Pool ARN des Bereichs General Settings (Allgemeine Einstellungen).

- d. Geben Sie für Token-Quelle als Header-Name **Authorization** ein, um das Identitäts- oder Zugriffstoken zu übergeben, das von Amazon Cognito bei erfolgreicher Anmeldung eines Benutzers zurückgegeben wird.
- e. Optional können Sie einen regulären Ausdruck im Feld Tokenvalidierung eingeben, um das aud-Feld (Zielgruppe) des Identitätstokens auszuwerten, bevor die Anfrage mit Amazon Cognito genehmigt wird. Beachten Sie, dass diese Validierung bei Verwendung eines Zugriffstoken die Anforderung zurückweist, da das Zugriffstoken das aud-Feld nicht enthält.
- f. Wählen Sie Genehmiger erstellen aus.

Erstellen Sie einen Amazon Cognito Cognito-Authorizer für eine REST-API mit CloudFormation

Sie können CloudFormation damit einen Amazon Cognito Cognito-Benutzerpool und einen Amazon Cognito Cognito-Autorisierer erstellen. Die CloudFormation Beispielvorlage macht Folgendes:

- Erstellen eines Amazon-Cognito-Benutzerpools. Der Client muss den Benutzer zunächst beim Benutzerpool anmelden und ein [Identitäts- oder Zugriffstoken](#) abrufen. Wenn Sie API-Methodenaufrufe mit Zugriffstoken autorisieren, stellen Sie bei der Konfiguration der App-Integration beim Benutzerpool sicher, dass die für einen bestimmten Ressourcenserver gewünschten benutzerdefinierten Bereiche eingerichtet werden.
- Erzeugt eine API-Gateway-API mit einer GET-Methode
- Erstellt einen Amazon-Cognito-Genehmiger, der den Authorization-Header als Token-Quelle verwendet.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      AccountRecoverySetting:
        RecoveryMechanisms:
          - Name: verified_phone_number
            Priority: 1
          - Name: verified_email
            Priority: 2
      AdminCreateUserConfig:
        AllowAdminCreateUserOnly: true
      EmailVerificationMessage: The verification code to your new account is {####}
      EmailVerificationSubject: Verify your new account
      SmsVerificationMessage: The verification code to your new account is {####}
      VerificationMessageTemplate:
        DefaultEmailOption: CONFIRM_WITH_CODE
        EmailMessage: The verification code to your new account is {####}
        EmailSubject: Verify your new account
        SmsMessage: The verification code to your new account is {####}
      UpdateReplacePolicy: Retain
      DeletionPolicy: Retain
  CogAuthorizer:
    Type: AWS::ApiGateway::Authorizer
    Properties:
      Name: CognitoAuthorizer
      RestApiId:
        Ref: Api
      Type: COGNITO_USER_POOLS
      IdentitySource: method.request.header.Authorization
      ProviderARNs:
        - Fn::GetAtt:
            - UserPool
            - Arn
  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyCogAuthApi
  ApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
```



```
    Ref: Api
  DependsOn:
    - CogAuthorizer
    - ApiGET
  ApiDeploymentStageprod:
    Type: AWS::ApiGateway::Stage
    Properties:
      RestApiId:
        Ref: Api
      DeploymentId:
        Ref: ApiDeployment
      StageName: prod
  ApiGET:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      ResourceId:
        Fn::GetAtt:
          - Api
          - RootResourceId
      RestApiId:
        Ref: Api
      AuthorizationType: COGNITO_USER_POOLS
      AuthorizerId:
        Ref: CogAuthorizer
      Integration:
        IntegrationHttpMethod: GET
        Type: HTTP_PROXY
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
  Outputs:
    ApiEndpoint:
      Value:
        Fn::Join:
          - ""
          - - https://
            - Ref: Api
            - .execute-api.
            - Ref: AWS::Region
            - "."
            - Ref: AWS::URLSuffix
            - /
            - Ref: ApiDeploymentStageprod
            - /
```

Integrationen für REST APIs in API Gateway

Nach dem Einrichten einer API-Methode müssen Sie diese in einen Endpunkt im Backend integrieren. Ein Backend-Endpunkt wird auch als Integrationsendpunkt bezeichnet und kann eine Lambda-Funktion, eine HTTP-Webseite oder eine AWS Serviceaktion sein.

Wie bei der API-Methode auch beinhaltet die API-Integration eine Integrationsanforderung eine Integrationsantwort. Die Integrationsanforderung kapselt eine HTTP-Anforderung vom Backend. Diese kann, muss sich aber nicht von der vom Client gesendeten Methodenanforderung unterscheiden. Die Integrationsantwort ist eine HTTP-Antwort, die die Ausgabe des Backends kapselt.

Das Einrichten einer Integrationsanforderung umfasst die folgenden Schritte: Konfiguration, wie vom Client gesendete Methodenanforderungen an das Backend übergeben werden, Konfiguration, wie die Anfragedaten gegebenenfalls in die Integrationsanforderungsdaten umgewandelt werden, Festlegen der aufzurufenden Lambda-Funktion, Festlegen des HTTP-Servers, an den die eingehende Anfrage weitergeleitet wird oder Festlegen, welche AWS -Service-Aktion aufgerufen wird.

Das Einrichten einer Integrationsantwort (ausschließlich auf Nicht-Proxy-Integrationen anwendbar) umfasst die folgenden Schritte: Konfiguration, wie das vom Backend zurückgegebene Ergebnis an eine Methodenantwort eines gegebenen Statuscodes übergeben wird, Konfiguration, wie die festgelegten Parameter der Integrationsantwort in vorkonfigurierte Parameter der Methodenantwort umgewandelt werden und Konfiguration, wie der Inhalt der Integrationsantwort gemäß den angegebenen Mapping-Vorlagen dem Inhalt der Methodenantwort zugeordnet wird.

Programmgesteuert wird eine Integrationsanforderung durch die Ressource [Integration](#) und eine Integrationsantwort durch die Ressource [IntegrationResponse](#) von Amazon API Gateway gekapselt.

Um eine Integrationsanforderung einzurichten, erstellen Sie eine [Integration](#)-Ressource und verwenden diese zum Konfigurieren der Integrations-Endpunkt-URL. Dann richten Sie die IAM-Berechtigungen für den Zugriff auf das Backend ein und legen die Mappings für die Umwandlung der eingehenden Anfragedaten vor der Übergabe an das Backend fest. Um eine Integrationsantwort für eine Nicht-Proxy-Integration einzurichten, erstellen Sie eine [IntegrationResponse](#)-Ressource und verwenden diese zum Festlegen der Zielmethodenantwort. Dann konfigurieren Sie, wie die Backend-Ausgabe der Methodenausgabe zugeordnet wird.

Themen

- [Integrationsanfrage in API Gateway einrichten](#)

- [Integrationsantwort in API Gateway einrichten](#)
- [Lambda-Integrationen für REST APIs in API Gateway](#)
- [HTTP-Integrationen für REST APIs in API Gateway](#)
- [Streamen Sie die Integrationsantwort für Ihre Proxy-Integrationen in API Gateway](#)
- [Private Integrationen für REST APIs in API Gateway](#)
- [Scheinintegrationen für REST APIs in API Gateway](#)

Integrationsanfrage in API Gateway einrichten

Führen Sie die folgenden erforderlichen und optionalen Schritte aus, um eine Integrationsanforderung einzurichten:

1. Wählen Sie einen Integrationstyp aus, um festzulegen, wie Methodenanforderungsdaten an das Backend übergeben werden.
2. Geben Sie für alle Integrationen außerdem dem Integrationstyp MOCK eine HTTP-Methode und die URI des gewählten Integrationsendpunkts an.
3. Für Integrationen mit Lambda-Funktionen und anderen AWS Serviceaktionen richten Sie eine IAM-Rolle mit den erforderlichen Berechtigungen ein, damit API Gateway das Backend in Ihrem Namen aufruft.
4. Richten Sie für Nicht-Proxy-Integrationen die erforderlichen Parameter-Mappings ein, um vordefinierte Methodenanforderungs-Parameter den entsprechenden Integrationsanforderungs-Parametern zuzuordnen.
5. Richten Sie für Nicht-Proxy-Integrationen die erforderlichen Inhalts-Mappings ein, um den Inhalt der eingehenden Methodenanforderung eines bestimmten Inhaltstyps gemäß der angegebenen Mapping-Vorlage zuzuordnen.
6. Geben Sie für Nicht-Proxy-Integrationen die Bedingung an, unter der die eingehenden Methodenanforderungsdaten unverändert an das Backend übergeben werden.
7. Legen Sie optional fest, wie Typumwandlungen für binäre Nutzlasten verarbeitet werden sollen.
8. Deklarieren Sie optional einen Cache-Namespace-Namen und Cache-Schlüsselparameter, um API-Caching zu aktivieren.

Zur Durchführung dieser Aufgaben müssen Sie eine API Gateway-[Integrations](#)-Ressource erstellen und entsprechende Eigenschaftswerte festlegen. Sie können dazu die API Gateway Gateway-Konsole, AWS CLI Befehle, ein AWS SDK oder die API Gateway Gateway-REST-API verwenden.

Themen

- [Grundlegende Aufgaben einer API-Integrationsanforderung](#)
- [Auswählen eines API Gateway-API-Integration](#)
- [Einrichten der Proxy-Integration mit einer Proxy-Ressource](#)
- [API-Integrationsanfrage über die API Gateway-Konsole einrichten](#)

Grundlegende Aufgaben einer API-Integrationsanforderung

Eine Integrationsanforderung ist eine HTTP-Anfrage, die API Gateway an das Backend sendet. Dabei werden die vom Client gesendeten Anfragedaten übergeben und gegebenenfalls umgewandelt. Die HTTP-Methode (oder Verb) und URI der Integrationsanforderung werden vom Backend, also dem Integrationsendpunkt, vorgegeben. Sie können, müssen aber nicht identisch mit der HTTP-Methode und URI der Methodenanforderung übereinstimmen.

Wenn eine Lambda-Funktion eine Datei zurückgibt, die von Amazon S3 abgerufen wurde, können Sie diesen Vorgang beispielsweise intuitiv als GET-Methodenanfrage für den Client bereitstellen, obwohl die entsprechende Integrationsanfrage festlegt, dass eine POST-Anfrage zum Aufruf der Lambda-Funktion verwendet wird. Bei einem HTTP-Endpunkt ist es wahrscheinlich, dass für die Methodenanforderung und die entsprechende Integrationsanforderung dasselbe HTTP-Verb verwendet wird. Dies ist jedoch nicht erforderlich. Sie können die folgende Methodenanforderung integrieren:

```
GET /{var}?query=value
Host: api.domain.net
```

Mit der folgenden Integrationsanforderung:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
  path: "{var}'s value",
  type: "value"
}
```

Als API-Entwickler können Sie beliebige HTTP-Verben und URIs für eine Methodenanforderung verwenden, die Ihre Anforderungen erfüllt. Dabei müssen jedoch die Anforderungen des Integrationsendpunkts erfüllt werden. Wenn die Methodenanforderungsdaten sich von den Daten der Integrationsanforderung unterscheiden, können Sie den Unterschied dadurch ausgleichen, dass Sie das Mapping der Methodenanforderungsdaten zu den Integrationsanforderungsdaten angeben.

In den vorherigen Beispielen werden die Werte für die Pfadvariable (`{var}`) und den Abfrageparameter (`query`) der GET-Methodenanforderung in Werte der Nutzlasteigenschaften der Integrationsanforderung `path` und `type` zugeordnet. Auch der Header und Inhalt der Anforderungsdaten können zugeordnet werden. Diese werden in [the section called "Parameterzuweisung"](#) beschrieben.

Beim Einrichten der HTTP- oder HTTP-Proxy-Integrationsanforderung weisen Sie die URL des Backend-HTTP-Endpunkts als URI-Wert der Integrationsanforderung zu. In der PetStore API hat die Methodenanforderung zum Abrufen einer Seite mit Haustieren beispielsweise den folgenden URI für die Integrationsanforderung:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Beim Einrichten der Lambda- oder Lambda-Proxy-Integration weisen Sie den Amazon-Ressourcennamen (ARN) für den Aufruf der Lambda-Funktion als URI-Wert der Integrationsanforderung zu. Der ARN hat das folgende Format:

```
arn:aws:apigateway:api-region:lambda:path/(/2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

Der Teil nach `arn:aws:apigateway:api-region:lambda:path/` (`/2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations`) ist der REST-API-URI-Pfad der Lambda-[Aufruf](#)-Aktion. Wenn Sie die API Gateway-Konsole zum Einrichten der Lambda-Integration verwenden, erstellt API Gateway den ARN und weist ihn der Integrations-URI zu, nachdem Sie aufgefordert wurden, den *lambda-function-name* aus einer Region auszuwählen.

Wenn die Integrationsanforderung mit einer anderen AWS Dienstaktion eingerichtet wird, ist der Integrationsanforderungs-URI ebenfalls ein ARN, ähnlich der Integration mit der Invoke Lambda-Aktion. Für die Integration mit der [GetBucket](#)Aktion von Amazon S3 ist die Integrationsanforderungs-URI beispielsweise ein ARN mit dem folgenden Format:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

Die Integrationsanforderungs-URI entspricht der Pfadkonvention zur Festlegung der Aktion, wobei *{bucket}* der Platzhalter für einen Bucket-Namen ist. Alternativ kann auf eine AWS Serviceaktion anhand ihres Namens verwiesen werden. Mit dem Aktionsnamen sieht die Integrationsanforderungs-URI für die GetBucket-Aktion für Amazon S3 wie folgt aus:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

In der aktionsbasierten Integrationsanforderungs-URI muss der Bucket-Name (*{bucket}*) im Inhalt der Integrationsanforderung (`{ Bucket: "{bucket}" }`) im Eingabeformat der GetBucket-Aktion festgelegt werden.

Für AWS Integrationen müssen Sie auch [Anmeldeinformationen](#) konfigurieren, damit API Gateway die integrierten Aktionen aufrufen kann. Sie können eine neue IAM-Rolle erstellen oder eine vorhandene IAM-Rolle auswählen, mit der API Gateway die Aktion aufruft, und dann die Rolle mit ihrem ARN angeben. Hier ein Beispiel für diesen ARN:

```
arn:aws:iam::account-id:role/iam-role-name
```

Die IAM-Rolle muss eine Richtlinie enthalten, in der die Berechtigung für die auszuführende Aktion gewährt wird. Außerdem muss API Gateway (in der Vertrauensbeziehung der Rolle) als vertrauenswürdige Entität deklariert sein, um die Rolle übernehmen zu können. Solche Berechtigungen können für die Aktion selbst gewährt werden. Sie werden als ressourcenbasierte Berechtigungen bezeichnet. Für die Lambda-Integration können Sie die Lambda-Aktion [addPermission](#) aufrufen, um die ressourcenbasierten Berechtigungen festzulegen, und dann `credentials` in der API Gateway-Integrationsanforderung auf Null festlegen.

Wir haben die Grundlagen der Integrationseinrichtung besprochen. Die erweiterten Einstellungen umfassen die Möglichkeit des Mapping der Methodenanforderungsdaten zu den Integrationsanforderungsdaten. Weitere Informationen finden Sie unter [the section called "Datentransformationen"](#).

Auswählen eines API Gateway-API-Integration

Die Wahl des API-Integrationstyps erfolgt gemäß dem verwendeten Typs des Integrationsendpunkts und der Übergabemethode von Daten an und von dem Integrationsendpunkt. Für eine Lambda-Funktion können Sie die Lambda-Proxy-Integration oder die benutzerdefinierte Lambda-

Integration verwenden. Für einen HTTP-Endpunkt können Sie die HTTP-Proxy-Integration oder die benutzerdefinierte HTTP-Integration verwenden. Für eine AWS Dienstaktion steht Ihnen nur die AWS Integration des Nicht-Proxy-Typs zur Verfügung. API Gateway unterstützt auch die Mock-Integration, bei der API Gateway als Integrationsendpunkt dient, um auf eine Methodenanfragen zu antworten.

Die benutzerdefinierte Lambda-Integration ist ein Sonderfall der AWS Integration, bei dem der Integrationsendpunkt der [funktionsaufrufenden Aktion des Lambda-Service](#) entspricht.

Programmgesteuert wählen Sie einen Integrationstyp durch Festlegen der Eigenschaft [type](#) für die Ressource [Integration](#) aus. Für die Lambda-Proxy-Integration ist der Wert `AWS_PROXY`. Für die benutzerdefinierte Lambda-Integration und alle anderen AWS -Integrationen lautet er `AWS`. Für die HTTP-Proxy-Integration und die HTTP-Integration ist der Wert `HTTP_PROXY` bzw. `HTTP`. Für die Pseudo-Integration ist der `type`-Wert `MOCK`.

Die Lambda-Proxy-Integration unterstützt eine vereinfachte Integrationseinrichtung mit einer einzigen Lambda-Funktion. Die Einrichtung ist einfach und kann um das Backend erweitert werden, ohne die vorhandene Einrichtung zu beeinträchtigen. Aus diesen Gründen wird sie für die Integration mit einer Lambda-Funktion dringend empfohlen.

Mit der benutzerdefinierten Lambda-Integration können Sie dagegen konfigurierte Mapping-Vorlagen für verschiedene Integrationsendpunkte mit vergleichbaren Anfragen für die Ein- und Ausgabedatenformate wiederverwenden. Die Einrichtung ist aufwändiger und wird für erweiterte Anwendungsszenarios empfohlen.

Ebenso ist die Integrationseinrichtung für die HTTP-Proxy-Integration optimiert und kann um das Backend erweitert werden, ohne die bestehende Einrichtung zu beeinträchtigen. Die benutzerdefinierte HTTP-Integration ist aufwändiger einzurichten, kann jedoch in Form von konfigurierten Mapping-Vorlagen für andere Integrationsendpunkte wiederverwendet werden.

In der folgenden Liste sind die unterstützten Integrationstypen zusammengefasst:

- **AWS:** Mit diesem Integrationstyp kann eine API AWS -Service-Aktionen festlegen. Bei der AWS-Integration müssen Sie sowohl die Integrationsanforderung als auch die Integrationsantwort konfigurieren und die erforderlichen Daten-Mappings von der Methodenanforderung zur Integrationsanforderung sowie von der Integrationsantwort zur Methodenantwort einrichten.
- **AWS_PROXY:** Mit diesem Integrationstyp lassen sich API-Methoden in die Lambda-Funktionsaufrufaktion integrieren. Die Einrichtung ist flexibel, vielseitig und optimiert. Die Integration stützt sich auf direkte Interaktionen zwischen dem Client und der integrierten Lambda-Funktion.

Bei diesem, auch als Lambda-Proxy-Integration bezeichneten Integrationstyp, legen Sie weder die Integrationsanforderung noch die Integrationsantwort fest. API Gateway übergibt die eingehende Anfrage vom Client als Eingabe an die Backend-Lambda-Funktion. Die integrierte Lambda-Funktion übernimmt die [Eingabe in diesem Format](#) und analysiert die Eingabe aus allen verfügbaren Quellen, darunter Anfrage-Header, URL-Pfadvariablen, Abfragezeichenfolge-Parameter und anwendbarer Inhalt. Die Funktion gibt das Ergebnis in diesem [Ausgabeformat](#) zurück.

Dies ist der bevorzugte Integrationstyp zum Aufrufen einer Lambda-Funktion über API Gateway und gilt nicht für andere AWS Dienstaktionen, einschließlich Lambda-Aktionen außer der funktionsaufrufenden Aktion.

- **HTTP:** Dieser Integrationstyp stellt eine API als HTTP-Endpunkte im Backend dar. Bei diesem, auch als benutzerdefinierte HTTP-Integration bezeichneten HTTP-Integrationstyp, müssen Sie sowohl die Integrationsanforderung als auch die Integrationsantwort konfigurieren. Sie müssen die erforderlichen Daten-Mappings von der Methodenanforderung zur Integrationsanforderung und von der Integrationsantwort zur Methodenantwort einrichten.
- **HTTP_PROXY:** Die HTTP-Proxy-Integration ermöglicht einem Client den Zugriff auf HTTP-Endpunkte des Backend über eine optimierte Integrationseinrichtung mit einer einzigen API-Methode. Sie legen keine Integrationsanforderung oder Integrationsantwort fest. Amazon API Gateway übergibt die eingehende Anfrage des Clients an den HTTP-Endpunkt und übergibt die ausgehende Antwort vom HTTP-Endpunkt an den Client.
- **MOCK:** Bei diesem Integrationstyp kann API Gateway eine Antwort zurückgeben, ohne die Anfrage weiter an das Backend zu senden. Dies ist für API-Test hilfreich, da Sie so die Integrationseinrichtung testen können, ohne dass Ihnen für die Nutzung des Backends Kosten entstehen, oder gemeinsam eine API entwickeln können.

Bei der gemeinsamen Entwicklung kann ein Team den Entwicklungsvorgang isolieren, indem es Simulationen von API-Komponenten anderer Teams über MOCK-Integrationen einrichtet. Dieser Integrationstyp wird auch verwendet, um CORS-Header zum Zulassen des CORS-Zugriffs durch die API-Methode zurückzugeben. Die API Gateway-Konsole integriert die `OPTIONS`-Methode zur Unterstützung von CORS sogar mit einer Mock-Integration. [Gateway-Antworten](#) sind weitere Beispiele für Pseudo-Integrationen.

Einrichten der Proxy-Integration mit einer Proxy-Ressource

Um eine Proxy-Integration in eine API Gateway-API mit einer [Proxy-Ressource](#) einzurichten, führen Sie die folgenden Aufgaben aus:

- Erstellen Sie eine Proxyressource mit der gierigen Pfadvariable `{proxy+}`.
- Legen Sie auf der Proxy-Ressource die Methode ANY fest.
- Integrieren Sie die Ressource und die Methode mithilfe des HTTP- oder Lambda-Integrationstyps mit einem Backend.

Note

Gierige Pfadvariablen, ANY-Methoden und Proxy-Integrationstypen sind unabhängige Funktionen, auch wenn sie häufig gemeinsam verwendet werden. Sie können eine bestimmte HTTP-Methode auf einer gierigen Ressource konfigurieren oder Nicht-Proxy-Integrationstypen auf eine Proxy-Ressource anwenden.

API Gateway setzt bestimmte Beschränkungen und Limitierungen durch, wenn Methoden mit einer Lambda-Proxy-Integration oder einer HTTP-API-Integration behandelt werden. Details hierzu finden Sie unter [the section called “Wichtige Hinweise”](#).

Note

Bei der Verwendung einer Proxy-Integration mit einem Pass-Through gibt API Gateway den Standard-Content-Type: `application/json`-Header zurück, wenn der Inhaltstyp eines Payloads nicht spezifiziert ist.

Eine Proxy-Ressource ist am leistungsstärksten, wenn sie mit der HTTP-Proxy-Integration oder der Lambda-Proxy-[Integration](#) in ein Backend integriert ist.

HTTP-Proxy-Integration mit einer Proxy-Ressource

Die HTTP-Proxy-Integration, die in der REST-API des API Gateways mit `HTTP_PROXY` festgelegt ist, dient der Integration einer Methodenanfrage mit einem HTTP-Backend-Endpunkt. Bei diesem Integrationstyp übergibt API Gateway die gesamte Anfrage und Antwort einfach zwischen dem Frontend und dem Backend. Dabei gelten bestimmte [Einschränkungen und Begrenzungen](#).

 Note

Die HTTP-Proxy-Integration unterstützt mehrwertige Header und Abfragezeichenfolgen.

Beim Anwenden der HTTP-Proxy-Integration auf eine Proxy-Ressource können Sie die API so einrichten, dass entweder ein Teil der oder die gesamte Endpunkthierarchie des HTTP-Backends mit einer einzigen Integrationseinrichtung bereitgestellt wird. Angenommen, das Backend der Website ist in mehrere Strukturknoten-Branches eines Stammknotens (`/site`) aufgeteilt: `/site/a0/a1/.../aN`, `/site/b0/b1/.../bM` usw. Wenn Sie die Methode ANY für eine Proxy-Ressource von `/api/{proxy+}` mit den Backend-Endpunkten mit den URL-Pfaden von `/site/{proxy}` integrieren, kann eine einzelne Integrationsanforderung alle HTTP-Vorgänge (GET, POST usw.) für beliebige von `[a0, a1, ..., aN, b0, b1, ..., bM, ...]` unterstützen. Wenn Sie eine Proxy-Integration stattdessen auf eine bestimmte HTTP-Methode anwenden, z. B. GET, funktioniert die resultierende Integrationsanforderung mit den angegebenen Vorgängen (d. h. GET) auf jedem dieser Backend-Knoten.

Lambda-Proxy-Integration mit einer Proxy-Ressource

Die Lambda-Proxy-Integration, in der API Gateway-REST-API mit `AWS_PROXY` festgelegt ist, dient der Integration einer Methodenanfrage mit einer Lambda-Funktion im Backend. Bei diesem Integrationstyp wendet API Gateway eine Standard-Mapping-Vorlage, um die gesamte Anfrage an die Lambda-Funktion zu senden, und wandelt die Ausgabe der Lambda-Funktion in HTTP-API-Antworten um.

Ebenso können Sie die Lambda-Proxy-Integration auf eine Proxy-Ressource von `/api/{proxy+}` anwenden, um eine einzelne Integration einzurichten, sodass eine Backend-Lambda-Funktion individuell auf Änderungen an einer der API-Ressourcen unter `/api` reagiert.

API-Integrationsanfrage über die API Gateway-Konsole einrichten

Die Einrichtung einer API-Methode definiert die Methode und beschreibt ihr Verhalten. Um eine Methode einzurichten, müssen Sie Folgendes angeben: eine Ressource, einschließlich des Roots (GET/POST), auf der die Methode bereitgestellt wird, eine HTTP-Methode (GET, POST usw.) und wie diese in das Ziel-Backend integriert wird. Die Methodenanforderung und -antwort geben die Regeln für den Austausch mit der aufrufenden App vor. Sie legen fest, welche Parameter die API empfangen kann und wie die Antwort aussieht.

In den folgenden Verfahren wird beschrieben, wie die API-Gateway-Konsole zum Erstellen einer Integrationsanforderung verwendet wird.

Themen

- [Einrichten einer Lambda-Integration](#)
- [Einrichten von HTTP-Integrationen](#)
- [Richten Sie eine AWS Serviceintegration ein](#)
- [Einrichten von Mock-Integrationen](#)

Einrichten einer Lambda-Integration

Verwenden Sie eine Lambda-Funktionsintegration, um Ihre API in eine Lambda-Funktion zu integrieren. Auf API-Ebene ist dies ein AWS-Integrationstyp, wenn Sie eine Nicht-Proxy-Integration erstellen, oder ein AWS_PROXY-Integrationstyp, wenn Sie eine Proxy-Integration erstellen.

Einrichten einer Integrationsmethode

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Für Integrationstyp wählen Sie Lambda-Funktion aus.
4. Um eine Lambda-Proxyintegration zu verwenden, aktivieren Sie Lambda proxy integration (Lambda-Proxyintegration). Weitere Informationen zum Erstellen von Lambda-Proxy-Integrationen finden Sie unter [the section called “ Lambda-Proxy-Integration kennenlernen ”](#).
5. Geben Sie für Lambda function (Lambda-Funktion) den Namen der Funktion ein.

Wenn Sie eine Lambda-Funktion in einer anderen Region als Ihrer API verwenden, wählen Sie die Region aus dem Dropdownmenü aus und geben Sie den Namen der Lambda-Funktion ein. Wenn Sie eine kontoübergreifende Lambda-Funktion verwenden, geben Sie die Funktion ARN ein.

6. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.

7. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 2 von [the section called “Bearbeiten einer Methodenanforderung in der Konsole”](#).

Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

8. Wählen Sie Methode erstellen aus.

Einrichten von HTTP-Integrationen

Verwenden Sie eine HTTP-Integration, um Ihre API in einen HTTP-Endpunkt zu integrieren. Auf der API-Ebene ist dies der HTTP Integrationstyp.

Einrichten von HTTP-Integrationen

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie unter Integrationstyp die Option HTTP aus.
4. Um eine HTTP-Proxyintegration zu verwenden, aktivieren Sie HTTP proxy integration (HTTP-Proxyintegration). Weitere Informationen zu HTTP-Proxyintegrationen finden Sie unter [the section called “HTTP-Proxy-Integrationen in API Gateway einrichten”](#).
5. Wählen Sie als HTTP method die HTTP-Methode aus, die am ehesten der Methode im HTTP-Backend entspricht.
6. Geben Sie als Endpoint URL (Endpunkt-URL) die URL des HTTP-Backends ein, das diese Methode verwenden soll.
7. Wählen Sie für Content handling (Inhaltsbehandlung) ein Verhalten bei der Inhaltsbehandlung aus.
8. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.
9. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und

konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 2 von [the section called “Bearbeiten einer Methodenanforderung in der Konsole”](#).

Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

10. Wählen Sie Methode erstellen aus.

Richten Sie eine AWS Serviceintegration ein

Verwenden Sie eine AWS Serviceintegration, um Ihre API direkt in einen AWS Dienst zu integrieren. Auf der API-Ebene ist dies der AWS Integrationstyp.

Gehen Sie zum Einrichten einer API Gateway wie folgt vor:

- Erstellen Sie eine neue Lambda-Funktion.
- Legen Sie eine Ressourcenberechtigung für die Lambda-Funktion fest.
- Führen Sie alle anderen Lambda-Serviceaktionen aus.

Sie müssen den AWS -Service wählen.

Um eine AWS Serviceintegration einzurichten

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie unter Integrationstyp die Option AWS -Service aus.
4. Wählen Sie unter AWS Region die AWS Region aus, die diese Methode zum Aufrufen der Aktion verwenden soll.
5. Wählen Sie für AWS Service den AWS Dienst aus, den Sie mit dieser Methode aufrufen möchten.
6. Geben Sie für AWS Subdomain die Subdomain ein, die AWS vom Service verwendet wird. Normalerweise bleibt dieses Feld leer. Einige AWS -Services können Subdomänen als Teil des Hosts unterstützen. Weitere Informationen über die Verfügbarkeit und gegebenenfalls Details finden Sie in der Service-Dokumentation.

7. Wählen Sie als HTTP method die der Aktion entsprechende HTTP-Methode aus. Informationen zum HTTP-Methodentyp finden Sie in der API-Referenzdokumentation für den AWS Dienst, den Sie für den Dienst ausgewählt haben.AWS
8. Wählen Sie unter Action type (Aktionstyp) entweder Use action name (Aktionsnamen verwenden) aus, um eine API-Aktion zu verwenden, oder Use path override (Pfadüberschreibung verwenden), um einen benutzerdefinierten Ressourcenpfad zu verwenden. Verfügbare Aktionen und benutzerdefinierte Ressourcenpfade finden Sie in der API-Referenzdokumentation für den AWS Dienst, den Sie für den AWS Dienst ausgewählt haben.
9. Geben Sie entweder einen Action name (Aktionsnamen) oder eine Path override (Pfadüberschreibung) ein.
10. Geben Sie unter Execution Role (Ausführungsrolle) den ARN der IAM-Rolle ein, die die Methode zum Aufrufen der Aktion verwendet.

Zum Erstellen einer IAM-Rolle können Sie die Anweisungen in [the section called “Schritt 1: Erstellen Sie die AWS Service-Proxy-Ausführungsrolle”](#) anpassen. Geben Sie eine Zugriffsrichtlinie mit der gewünschten Anzahl von Aktions- und Ressourcenanweisungen an. Weitere Informationen finden Sie unter [the section called “Funktionsweise von Amazon API Gateway mit IAM”](#).

Die Syntax der Aktions- und Ressourcenanweisungen finden Sie in der Dokumentation des AWS Dienstes, den Sie für den AWS Dienst ausgewählt haben.

Für die Vertrauensbeziehung der IAM-Rolle geben Sie Folgendes an, wodurch API Gateway in der Lage ist, im Namen Ihres AWS -Kontos aktiv zu werden:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
}
```

11. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.
12. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 2 von [the section called “Bearbeiten einer Methodenanforderung in der Konsole”](#).

Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

13. Wählen Sie Methode erstellen aus.

Einrichten von Mock-Integrationen

Wählen Sie Mock aus, wenn API Gateway als Backend fungieren soll, um statische Antworten zurückzugeben. Auf der API-Ebene ist dies der MOCK Integrationstyp. Normalerweise können Sie die MOCK Integration verwenden, wenn Ihre API noch nicht finalisiert ist, Sie aber API-Antworten erzeugen möchten, um abhängigen Teams Tests zu ermöglichen. Für die OPTION-Methode legt API Gateway die MOCK-Integration als Standard fest, um CORS-fähige Header für die angewandte API-Ressource zurückzugeben.

Einrichten einer Integrationsmethode

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie unter Integrationstyp die Option Mock aus.
4. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 2 von [the section called “Bearbeiten einer Methodenanforderung in der Konsole”](#).

Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

5. Wählen Sie Methode erstellen aus.

Integrationsantwort in API Gateway einrichten

Bei einer Nicht-Proxy-Integration müssen Sie mindestens eine Integrationsantwort einrichten und als Standardantwort festlegen, um das Ergebnis vom Backend an den Client zu übergeben. Sie können das Ergebnis unverändert übergeben oder die Integrationsantwortdaten bei unterschiedlichem Format in die Methodenantwortdaten umwandeln.

Bei einer Proxy-Integration leitet API Gateway die Backend-Ausgabe automatisch als HTTP-Antwort an den Client weiter. Sie legen weder Integrationsantwort noch Methodenantwort fest.

Führen Sie die folgenden erforderlichen und optionalen Schritte aus, um eine Integrationsantwort einzurichten:

1. Geben Sie einen HTTP-Statuscode einer Methodenantwort an, der die Integrationsantwortdaten zugeordnet sind. Diese Information ist erforderlich.
2. Definieren Sie einen regulären Ausdruck, um Backend-Ausgaben als Integrationsantwort auszuwählen. Wenn Sie hier keine Angaben machen, wird als Antwort die Standardantwort verwendet, die immer dann verwendet wird, wenn keine anderweitige Antwort konfiguriert ist.
3. Deklarieren Sie gegebenenfalls Mappings bestehend aus Schlüssel-Wert-Paaren, um angegebene Integrationsantwortparameter den gegebenen Methodenantwortparametern zuzuordnen.
4. Fügen Sie gegebenenfalls Inhalts-Mapping-Vorlagen hinzu, um vorhandene Integrationsantwortnutzlasten in angegebene Methodenantwortnutzlasten umzuwandeln.
5. Legen Sie gegebenenfalls fest, wie Typumwandlungen für binäre Nutzlasten verarbeitet werden sollen.

Die Integrationsantwort ist eine HTTP-Antwort, die die Backend-Antwort kapselt. Bei HTTP-Endpunkten ist die Backend-Antwort eine HTTP-Antwort. Der Statuscode der Integrationsantwort kann den vom Backend zurückgegebenen Statuscode übernehmen. Der Inhalt der Integrationsantwort ist die vom Backend zurückgegebene Nutzlast. Bei einem Lambda-Endpunkt ist die Backend-Antwort die von der Lambda-Funktion zurückgegebene Ausgabe. Bei der Lambda-Integration wird die Ausgabe der Lambda-Funktion als 200 OK-Antwort zurückgegeben. Die Nutzlast kann das Ergebnis als JSON-Daten enthalten, einschließlich einer JSON-Zeichenfolge

oder einem JSON-Objekt oder einer Fehlermeldung als JSON-Objekt. Sie können der Eigenschaft [selectionPattern](#) einen regulären Ausdruck zuweisen, um eine Fehlerantwort einer passenden HTTP-Fehlerantwort zuzuordnen. Weitere Informationen über die Fehlerantwort der Lambda-Funktion finden Sie unter [Lambda-Fehler in API Gateway behandeln](#). Bei der Lambda-Proxy-Integration muss die Lambda-Funktion eine Ausgabe im folgenden Format zurückgeben:

```
{
  statusCode: "...",           // a valid HTTP status code
  headers: {
    custom-header: "..."     // any API-specific custom header
  },
  body: "...",                // a JSON string.
  isBase64Encoded: true|false // for binary support
}
```

Es besteht keine Notwendigkeit, die Antwort der Lambda-Funktion der richtigen HTTP-Antwort zuzuordnen.

Um das Ergebnis an den Client zurückzugeben, richten Sie die Integrationsantwort so ein, dass die Endpunktantwort unverändert an die entsprechende Methodenantwort übergeben wird. Alternativ können Sie die Endpunktantwortdaten auch den Methodenantwortdaten zuordnen. Es können folgende Antwortdaten zugeordnet werden: der Antwortstatuscode, die Parameter des Antwort-Headers und der Antwortinhalt. Wenn keine Methodenantwort für den zurückgegebenen Statuscode definiert ist, gibt API Gateway einen 500-Fehler zurück. Weitere Informationen finden Sie unter [Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway](#).

Lambda-Integrationen für REST APIs in API Gateway

Sie können eine API-Methode mit einer Lambda-Funktion integrieren, indem Sie eine Lambda-Proxy-Integration oder eine nicht-proxy (benutzerdefinierte) Lambda-Integration verwenden.

Bei der Lambda-Proxy-Integration ist die erforderliche Einrichtung einfach. Legen Sie die HTTP-Methode der Integration auf POST und den Integrationsendpunkt-URI auf den ARN der Lambda-Funktion-Aufrufsaktion einer bestimmten Lambda-Funktion fest. Gewähren Sie dann API Gateway die Berechtigung zum Aufrufen der Lambda-Funktion in Ihrem Namen.

In der Lambda-Nicht-Proxy-Integration müssen Sie zusätzlich zu den Einrichtungsschritten der Proxy-Integration angeben, wie die eingehenden Anfragedaten der Integrationsanforderung und die resultierenden Integrationsantwortdaten der Methodenantwort zugeordnet werden.

Themen

- [Lambda-Proxy-Integrationen in API Gateway](#)
- [Benutzerdefinierte Lambda-Integrationen in API Gateway einrichten](#)
- [Asynchronen Aufruf der Backend-Lambda-Funktion einrichten](#)
- [Lambda-Fehler in API Gateway behandeln](#)

Lambda-Proxy-Integrationen in API Gateway

Im folgenden Abschnitt wird gezeigt, wie Sie eine Lambda-Proxy-Integration verwenden.

Themen

- [API Gateway Lambda-Proxy-Integration kennenlernen](#)
- [Support für mehrwertige Header- und Abfragezeichenfolgenparameter](#)
- [Eingabeformat einer Lambda-Funktion für die Proxy-Integration](#)
- [Ausgabeformat einer Lambda-Funktion für die Proxy-Integration](#)
- [Richten Sie die Lambda-Proxyintegration für API Gateway mit dem ein AWS CLI](#)
- [Proxy-Ressource mit einer Lambda-Proxy-Integration mithilfe einer OpenAPI-Definition einrichten](#)

API Gateway Lambda-Proxy-Integration kennenlernen

Die Amazon API Gateway-Lambda-Proxy-Integration ist ein einfacher, leistungsstarker und schneller Mechanismus zur Erstellung einer API mit der Einrichtung einer einzigen API-Methode. Die Lambda-Proxy-Integration ermöglicht es dem Client, eine einzelne Lambda-Funktion im Backend aufzurufen. Die Funktion greift auf viele Ressourcen oder Funktionen anderer AWS Dienste zu, einschließlich des Aufrufs anderer Lambda-Funktionen.

Bei der Lambda-Proxy-Integration übergibt API Gateway beim Senden einer API-Anfrage ein [Ereignisobjekt](#) an die integrierte Lambda-Funktion. Allerdings wird die Reihenfolge der Anfrageparameter nicht beibehalten. Diese [Anforderungsdaten](#) umfassen die Abfrage-Header, Parameter für Abfragezeichenfolgen, die URL-Pfadvariablen, die Nutzlast und API-Konfigurationsdaten. Die Konfigurationsdaten können den aktuellen Bereitstellungsstufennamen, Stufenvariablen, die Benutzeridentität oder den Autorisierungskontext (falls vorhanden) umfassen. Die Backend-Lambda-Funktion analysiert die eingehenden Anfragedaten, um die eigene Antwort zu bestimmen. Damit API Gateway die Lambda-Ausgabe als API-Antwort an den Client weiterleiten kann, muss die Lambda-Funktion das Ergebnis in [diesem Format](#) zurückgeben.

Da API Gateway für die Lambda-Proxy-Integration zwischen dem Client und der Backend-Lambda-Funktion nur unwesentlich eingreift, können sich der Client und die integrierte Lambda-Funktion an Änderungen in der jeweils anderen Funktion anpassen, ohne die bestehende Integrationseinrichtung der API zu stören. Hierzu muss der Client den Anwendungsprotokollen der Backend-Lambda-Funktion folgen.

Sie können eine Lambda-Proxy-Integration für jede API-Methode einrichten. Aber eine Lambda-Proxy-Integration ist leistungstärker, wenn sie über eine allgemeine Proxy-Ressource für eine API-Methode konfiguriert wird. Die allgemeine Proxy-Ressource kann durch eine spezielle Vorlagenpfadvariable `{proxy+}`, den Catch-All-ANY-Methodenplatzhalter oder beides angegeben werden. Der Client kann die Eingabe an die Backend-Lambda-Funktion in der eingehenden Anfrage als Anfrageparameter oder als entsprechende Nutzlast übergeben. Zu den Anforderungsparametern gehören Header-, URL-Pfadvariablen, Abfragezeichenfolgenparameter und die entsprechende Nutzlast. Die integrierte Lambda-Funktion prüft alle Eingangsquellen vor ihrer Verarbeitung und der Antwort an den Client mit sinnvollen Fehlermeldungen, falls die erforderliche Eingabe fehlt.

Beim Aufrufen einer mit der allgemeinen HTTP-Methode ANY und der allgemeinen Ressource `{proxy+}` integrierten API-Methode sendet der Client eine Anforderung mit einer bestimmten HTTP-Methode statt mit ANY. Der Client gibt zudem einen bestimmten URL-Pfad statt `{proxy+}` an und umfasst alle erforderlichen Header, Abfragezeichenfolgenparameter oder eine anwendbare Nutzlast.

Die folgende Liste enthält eine Zusammenfassung des Laufzeitverhaltens verschiedener API-Methoden bei der Lambda-Proxy-Integration:

- ANY `/ {proxy+}`: Der Client muss eine bestimmte HTTP-Methode auswählen, eine bestimmte Ressourcenpfadhierarchie festlegen und kann alle Header, Parameter für Abfragezeichenfolgen und den Payload für die Eingabe an die integrierte Lambda-Funktion festlegen.
- ANY `/res`: Der Client muss eine bestimmte HTTP-Methode auswählen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabe an die integrierte Lambda-Funktion festlegen.
- GET|POST|PUT|... `/ {proxy+}`: Der Client kann eine bestimmte Ressourcenpfadhierarchie festlegen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabe an die integrierte Lambda-Funktion festlegen.
- GET|POST|PUT|... `/res/{path}/...`: Der Client muss ein bestimmtes Pfadsegment (für die `{path}`-Variable) festlegen und kann alle Anfrage-Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabedaten an die integrierte Lambda-Funktion festlegen.

- GET|POST|PUT|... /res: Der Client kann bestimmte Anfrage-Header auswählen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabedaten an die integrierte Lambda-Funktion festlegen.

Sowohl die Proxy-Ressource {proxy+} und die benutzerdefinierte Ressource {custom} werden als Vorlagenpfadvariablen ausgedrückt. Allerdings kann {proxy+} auf alle Ressourcen in einer Pfadhierarchie verweisen, während sich {custom} nur auf ein bestimmtes Pfadsegment bezieht. Beispielsweise könnte ein Lebensmittelgeschäft sein Online-Produktsortiment nach Abteilungsnamen, Kategorien und Produktarten organisieren. Die Website des Lebensmittelgeschäfts kann dann verfügbare Produkte über die folgenden Vorlagenpfadvariablen darstellen: /{department}/{produce-category}/{product-type}. Zum Beispiel werden Äpfel mit /produce/fruit/apple und Karotten mit /produce/vegetables/carrot dargestellt. Es kann auch /{proxy+} als Platzhalter für eine beliebige Abteilung, eine beliebige Kategorie oder einen Produkttyp verwendet werden, nach dem ein Kunde im Online-Shop suchen kann. Beispielsweise kann /{proxy+} auf eines der folgenden Elemente verweisen:

- /produce
- /produce/fruit
- /produce/vegetables/carrot

Um Kunden nach den verfügbaren Produkten, der Kategorie und der zugeordneten Abteilung suchen zu lassen, können Sie eine einzelne GET /{proxy+}-Methode mit Leseberechtigungen bereitstellen. Um es einem Supervisor zu ermöglichen, das Inventar der produce Abteilung zu aktualisieren, können Sie auf ähnliche Weise eine einzige Methode einrichten: PUT /produce/{proxy+} mit read/write Berechtigungen. Um es einem Kassierer zu ermöglichen, die Gesamtsumme eines Gemüses zu aktualisieren, können Sie eine POST /produce/vegetables/{proxy+} Methode mit read/write Berechtigungen einrichten. Damit ein Filialleiter alle möglichen Aktionen für jedes verfügbare Produkt ausführen kann, kann der Onlineshop-Entwickler die ANY /{proxy+} Methode mit read/write entsprechenden Berechtigungen verfügbar machen. In jedem Fall muss der Kunde oder der Mitarbeiter zur Laufzeit ein bestimmtes Produkt eines bestimmten Typs in einer bestimmten Abteilung, eine bestimmte Kategorie in einer bestimmten Abteilung, oder eine bestimmte Abteilung auswählen.

Weitere Informationen zum Einrichten der API Gateway-Proxy-Integrationen finden Sie unter [Einrichten der Proxy-Integration mit einer Proxy-Ressource](#).

Die Proxy-Integration erfordert, dass der Client über detaillierte Kenntnisse der Backend-Anforderungen verfügt. Um eine optimale App-Leistung und Benutzererfahrung sicherzustellen, muss der Backend-Entwickler die Anforderungen des Backends klar an den Client-Entwickler kommunizieren. Er muss einen robusten Feedback-Mechanismus für nicht erfüllte Anforderungen bereitstellen.

Support für mehrwertige Header- und Abfragezeichenfolgenparameter

API Gateway unterstützt mehrere gleichnamige Header und Abfragezeichenfolgenparameter. Mehrwertige Header sowie einwertige Header und Parameter können in denselben Anforderungen und Antworten kombiniert werden. Weitere Informationen erhalten Sie unter [Eingabeformat einer Lambda-Funktion für die Proxy-Integration](#) und [Ausgabeformat einer Lambda-Funktion für die Proxy-Integration](#).

Eingabeformat einer Lambda-Funktion für die Proxy-Integration

Bei der Lambda-Proxy-Integration ordnet API Gateway die gesamte Client-Anforderung dem event-Eingabeparameter der Backend-Lambda-Funktion zu. Das folgende Beispiel zeigt die Struktur eines Ereignisses, das API Gateway an eine Lambda-Proxy-Integration sendet.

In diesem Beispiel gehen wir davon aus, dass der Aufruf an API Gateway folgendermaßen erfolgte:

```
curl 'https://a1b2c3.execute-api.us-east-1.amazonaws.com/my/path?
parameter1=value1&parameter2=value1&parameter2=value2&parameter3=value1,value2'
-H 'header1: value1' -H 'header2: value1' -H 'header2: value2' -H 'header3:
value1,value2'
```

Die Ausgabe sollte wie folgt aussehen:

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2",
    "header3": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": ["value1"],
    "header2": ["value1","value2"],
    "header3": ["value1,value2"]
  }
}
```

```

},
"queryStringParameters": {
  "parameter1": "value1",
  "parameter2": "value2",
  "parameter3": "value1,value2"
},
"multiValueQueryStringParameters": {
  "parameter1": ["value1"],
  "parameter2": ["value1","value2"],
  "parameter3": ["value1,value2"]
},
"requestContext": {
  "accountId": "123456789012",
  "apiId": "id",
  "authorizer": {
    "claims": null,
    "scopes": null
  },
  "domainName": "id.execute-api.us-east-1.amazonaws.com",
  "domainPrefix": "id",
  "extendedRequestId": "request-id",
  "httpMethod": "GET",
  "identity": {
    "accessKey": null,
    "accountId": null,
    "caller": null,
    "cognitoAuthenticationProvider": null,
    "cognitoAuthenticationType": null,
    "cognitoIdentityId": null,
    "cognitoIdentityPoolId": null,
    "principalOrgId": null,
    "sourceIp": "IP",
    "user": null,
    "userAgent": "user-agent",
    "userArn": null,
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  }
}

```

```
    }  
  },  
  "path": "/my/path",  
  "protocol": "HTTP/1.1",  
  "requestId": "id=",  
  "requestTime": "04/Mar/2020:19:15:17 +0000",  
  "requestTimeEpoch": 1583349317135,  
  "resourceId": null,  
  "resourcePath": "/my/path",  
  "stage": "$default"  
},  
"pathParameters": null,  
"stageVariables": null,  
"body": "Hello from Lambda!",  
"isBase64Encoded": false  
}
```

Note

Wählen Sie im Bereich

- Der `headersSchlüssel` kann nur einwertige Header enthalten
- Der `multiValueHeadersSchlüssel` kann mehrwertige Header sowie einwertige Header enthalten.
- Wenn Sie Werte sowohl für `headers` als auch für `multiValueHeaders` angeben, führt API Gateway diese in einer einzigen Liste zusammen. Wenn in beiden das gleiche Schlüssel / Wert-Paar angegeben ist, werden nur die Werte von in der `multiValueHeaders` zusammengeführten Liste angezeigt.

In der Eingabe an die Backend-Lambda-Funktion ist das `requestContext`-Objekt eine Zuordnung von Schlüssel-Werte-Paaren. In jedem Paar ist der Schlüssel der Name einer [\\$context](#)-Variableneigenschaft, und der Wert ist der Wert dieser Eigenschaft. API Gateway kann der Zuordnung u. U. neue Schlüssel hinzufügen.

Abhängig von den aktivierten Funktionen kann die Zuweisung `requestContext` von API zu API unterschiedlich sein. Im obigen Beispiel ist z. B. kein Autorisierungstyp angegeben, sodass keine `$context.authorizer.*`- oder `$context.identity.*`-Eigenschaften vorhanden sind. Wenn ein Autorisierungstyp angegeben wird, bewirkt dies, dass API Gateway

autorisierte Benutzerinformationen wie folgt in einem `requestContext.identity`-Objekt an den Integrationsendpunkt weiterleitet:

- Wenn der Autorisierungstyp `AWS_IAM` ist, enthalten die autorisierten Benutzerinformationen `$context.identity.*`-Eigenschaften.
- Wenn der Autorisierungstyp `COGNITO_USER_POOLS` ist (Amazon Cognito-Genehmiger), enthalten die autorisierten Benutzerinformationen die Eigenschaften `$context.identity.cognito*` und `$context.authorizer.claims.*`.
- Wenn der Autorisierungstyp `CUSTOM` (Lambda-Genehmiger) ist, enthalten die autorisierten Benutzerinformationen `$context.authorizer.principalId`- und andere relevante `$context.authorizer.*`-Eigenschaften.

Ausgabeformat einer Lambda-Funktion für die Proxy-Integration

Bei der Lambda-Proxy-Integration erfordert API Gateway, dass die Backend-Lambda-Funktion die Ausgabe gemäß dem folgenden JSON-Format zurückgibt:

```
{
  "isBase64Encoded": true/false,
  "statusCode": httpStatusCode,
  "headers": { "headerName": "headerValue", ... },
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "body": "..."
}
```

In der Konsolenausgabe:

- Die Schlüssel `headers` und `multiValueHeaders` können nicht angegeben werden, wenn keine zusätzlichen Antwortheader zurückgegeben werden sollen.
- Der `headers`-Schlüssel kann nur einwertige Header enthalten
- Der `multiValueHeaders`-Schlüssel kann mehrwertige Header sowie einwertige Header enthalten. Sie können den `multiValueHeaders`-Schlüssel verwenden, um alle zusätzlichen Header anzugeben, einschließlich einzelner Werte.
- Wenn Sie Werte sowohl für `headers` als auch für `multiValueHeaders` angeben, führt API Gateway diese in einer einzigen Liste zusammen. Wenn in beiden das gleiche Schlüssel / Wert-Paar angegeben ist, werden nur die Werte von in der `multiValueHeaders`-zusammengeführten Liste angezeigt.

Um CORS für die Lambda-Proxy-Integration zu aktivieren, müssen Sie der Ausgabe headers `Access-Control-Allow-Origin:domain-name` hinzufügen. `domain-name` kann `*` für einen beliebigen Domännennamen sein. Die Ausgabe body wird als Nutzlast der Methodenanforderung an das Frontend angeordnet. Wenn body ein binärer Blob ist, können Sie ihn als Base64-kodierte Zeichenfolge codieren, indem Sie `isBase64Encoded` auf `true` festlegen und `*/*` als Binary Media Type (Binärer Medientyp) konfigurieren. Andernfalls können Sie den Wert auf `false` festlegen oder keinen Wert angeben.

 Note

Weitere Informationen zur Aktivierung der Unterstützung von Binärdateien finden Sie unter [Binärunterstützung über die API Gateway-Konsole aktivieren](#). Eine Beispielfunktion für Lambda finden Sie unter [Rückgabe binärer Medien aus einer Lambda-Proxy-Integration in API Gateway](#).

Wenn die Funktionsausgabe ein anderes Format hat, gibt API Gateway eine 502 Bad Gateway-Fehlerantwort zurück.

Um eine Antwort in einer Lambda-Funktion in Node.js zurückzugeben, können Sie Befehle wie den folgenden verwenden:

- Um ein erfolgreiches Ergebnis zurückzugeben, rufen Sie `callback(null, {"statusCode": 200, "body": "results"})`.
- Rufen Sie zum Auslösen einer Ausnahme `callback(new Error('internal server error'))`.
- Bei einem Fehler auf Clientseite, z. B. wenn ein erforderlicher Parameter fehlt, können Sie `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` aufrufen, um den Fehler zurückzugeben, ohne eine Ausnahme auszulösen.

In einer Lambda-async-Funktion in Node.js würde die entsprechende Syntax so lauten:

- Um ein erfolgreiches Ergebnis zurückzugeben, rufen Sie `return {"statusCode": 200, "body": "results"}`.
- Rufen Sie zum Auslösen einer Ausnahme `throw new Error("internal server error")`.

- Bei einem Fehler auf Clientseite, z. B. wenn ein erforderlicher Parameter fehlt, können Sie `return {"statusCode": 400, "body": "Missing parameters of ..."}` aufrufen, um den Fehler zurückzugeben, ohne eine Ausnahme auszulösen.

Richten Sie die Lambda-Proxyintegration für API Gateway mit dem ein AWS CLI

In diesem Abschnitt zeigen wir Ihnen, wie Sie mithilfe von AWS CLI eine API mit der Lambda-Proxy-Integration einrichten. Detaillierte Anweisungen zur Verwendung der API Gateway-Konsole zur Konfiguration einer Proxy-Ressource mit der Lambda-Proxy-Integration finden Sie unter [Tutorial: Erstellen einer REST-API mit einer Lambda-Proxy-Integration](#).

Als Beispiel verwenden wir die folgende Lambda-Beispielfunktion als Backend der API:

```
export const handler = async(event, context) => {
  console.log('Received event:', JSON.stringify(event, null, 2));
  var res = {
    "statusCode": 200,
    "headers": {
      "Content-Type": "*/*"
    }
  };
  var greeter = 'World';
  if (event.greeter && event.greeter !== "") {
    greeter = event.greeter;
  } else if (event.body && event.body !== "") {
    var body = JSON.parse(event.body);
    if (body.greeter && body.greeter !== "") {
      greeter = body.greeter;
    }
  } else if (event.queryStringParameters && event.queryStringParameters.greeter && event.queryStringParameters.greeter !== "") {
    greeter = event.queryStringParameters.greeter;
  } else if (event.multiValueHeaders && event.multiValueHeaders.greeter && event.multiValueHeaders.greeter !== "") {
    greeter = event.multiValueHeaders.greeter.join(" and ");
  } else if (event.headers && event.headers.greeter && event.headers.greeter !== "") {
    greeter = event.headers.greeter;
  }
  res.body = "Hello, " + greeter + "!";
  return res
};
```

Vergleicht man dies mit der benutzerdefinierten Lambda-Integrationseinrichtung unter [the section called “ Benutzerdefinierte Lambda-Integrationen einrichten ”](#), kann die Eingabe für diese Lambda-Funktion in den Anfrageparametern und im Text definiert werden. Sie haben mehr Spielraum, um dem Client die Weitergabe der gleichen Eingabedaten zu ermöglichen. Hier kann der Client den Namen von "greeter" als Abfragezeichenfolgeparameter, Kopfzeile oder Texteigenschaft übergeben. Die Funktion kann außerdem die benutzerdefinierte Lambda-Integration unterstützen. Die API-Einrichtung ist einfacher. Sie müssen keine Methoden- oder Integrationsantwort konfigurieren.

So richten Sie eine Lambda-Proxyintegration mit dem AWS CLI

1. Verwenden Sie den folgenden [create-rest-api](#)Befehl, um eine API zu erstellen:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "name": "HelloWorldProxy (AWS CLI)",
  "id": "te6si5ach7",
  "rootResourceId" : "krznpq9xpg",
  "createdDate": 1508461860
}
```

In diesem Beispiel verwenden Sie durchgängig die API id (te6si5ach7) und die rootResourceId (krznpq9xpg).

2. Verwenden Sie den folgenden [create-resource](#)-Befehl, um eine API-Gateway-[Ressource](#) mit dem Namen /greeting zu erstellen:

```
aws apigateway create-resource \
  --rest-api-id te6si5ach7 \
  --parent-id krznpq9xpg \
  --path-part {proxy+}
```

Die Ausgabe sieht wie folgt aus:

```
{
  "path": "/{proxy+}",
  "pathPart": "{proxy+}",
  "id": "2jf6xt",
```

```
"parentId": "krznpq9xpg"
}
```

Sie verwenden den `id`-Wert der Ressource `{proxy+}` (`2jf6xt`), um im nächsten Schritt eine Methode für die Ressource `/ {proxy+}` zu erstellen.

3. Verwenden Sie die folgende [put-method](#), um eine ANY-Methodenanforderung von ANY `/ {proxy+}` zu erstellen:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method ANY \
  --authorization-type "NONE"
```

Die Ausgabe sieht wie folgt aus:

```
{
  "apiKeyRequired": false,
  "httpMethod": "ANY",
  "authorizationType": "NONE"
}
```

Diese API-Methode gestattet dem Client, einen Gruß über die Lambda-Funktion am Backend zu empfangen oder zu senden.

4. Verwenden Sie den folgenden [put-integration](#)-Befehl, um die Integration der ANY `/ {proxy+}`-Methode mit einer Lambda-Funktion mit dem Namen `HelloWorld` einzurichten. Diese Funktion beantwortet die Anforderung mit der Nachricht `"Hello, {name}!"`, sofern der `greeter`-Parameter angegeben wurde, oder `"Hello, World!"`, wenn kein Abfragezeichenfolgeparameter festgelegt wurde.

```
aws apigateway put-integration \
  --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method ANY \
  --type AWS_PROXY \
  --integration-http-method POST \
  --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
  --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

⚠ Important

Für Lambda-Integrationen müssen Sie entsprechend der [Spezifikation der Lambda-Service-Aktion für Funktionsaufrufe](#) die HTTP-Methode POST für die Integrationsanfrage verwenden. Die IAM-Rolle `apigAwsProxyRole` muss über Richtlinien verfügen, die es dem `apigateway`-Service erlauben, Lambda-Funktionen aufzurufen. Weitere Informationen zu IAM-Berechtigungen finden Sie unter [the section called “API Gateway-Berechtigungsmo-
dell für den Aufruf einer API”](#).

Die Ausgabe sieht wie folgt aus:

```
{
  "passthroughBehavior": "WHEN_NO_MATCH",
  "cacheKeyParameters": [],
  "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:1234567890:function:HelloWorld/invocations",
  "httpMethod": "POST",
  "cacheNamespace": "v vom7n",
  "credentials": "arn:aws:iam::1234567890:role/apigAwsProxyRole",
  "type": "AWS_PROXY"
}
```

Anstatt eine IAM-Rolle für `credentials` bereitzustellen, können Sie den Befehl [add-permission](#) verwenden, um ressourcenbasierte Berechtigungen hinzuzufügen. Dies erledigt die API Gateway-Konsole.

5. Verwenden Sie den folgenden [create-deployment](#)-Befehl, um die API für eine test-Stufe bereitzustellen:

```
aws apigateway create-deployment \
  --rest-api-id te6si5ach7 \
  --stage-name test
```

6. Testen Sie die API mithilfe der folgenden cURL-Befehle in einem Terminal.

API-Aufruf mit dem Abfragezeichenfolgeparameter `?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?greeter=jane'
```

API-Aufruf mit einem Header-Parameter greeter:jane:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -H 'greeter: jane'
```

API-Aufruf mit einem Textkörper-Parameter {"greeter":"jane"}:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -d '{ "greeter": "jane" }'
```

In allen Fällen ist die Ausgabe eine Antwort mit dem folgenden Antworttext: 200

```
Hello, jane!
```

Proxy-Ressource mit einer Lambda-Proxy-Integration mithilfe einer OpenAPI-Definition einrichten

Erstellen Sie zum Einrichten einer Proxy-Ressource mit dem Lambda-Proxy-Integrationstyp eine API-Ressource mit einem Greedy-Pfadparameter (z. B. /parent/{proxy+}) und integrieren Sie diese Ressource mit einem Lambda-Funktions-Backend (z. B. arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource) für die Methode ANY. Der Greedy-Pfadparameter muss am Ende des API-Ressourcenpfads stehen. Wie bei einer Nicht-Proxy-Ressource können Sie die Proxy-Ressource einrichten, indem Sie die API Gateway-Konsole verwenden, eine OpenAPI-Definitionsdatei importieren oder die API Gateway-REST-API direkt aufrufen.

Die folgende OpenAPI-API-Definitionsdatei zeigt ein Beispiel für eine API mit einer Proxyressource, die mit der Lambda-Funktion SimpleLambda4ProxyResource integriert wird.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
```

```

    "version": "2016-09-12T17:50:37Z",
    "title": "ProxyIntegrationWithLambda"
  },
  "paths": {
   ("/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "cacheNamespace": "roq9wj",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "aws_proxy"
        }
      }
    }
  },
  "servers": [
    {
      "url": "https://gy415nuibc.execute-api.us-east-1.amazonaws.com/{basePath}",
      "variables": {
        "basePath": {
          "default": "/testStage"
        }
      }
    }
  ]
}

```

```

    }
  }
]
}

```

OpenAPI 2.0

```

{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T17:50:37Z",
    "title": "ProxyIntegrationWithLambda"
  },
  "host": "gy415nuibc.execute-api.us-east-1.amazonaws.com",
  "basePath": "/testStage",
  "schemes": [
    "https"
  ],
  "paths": {
   ("/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/invocations",
          "passthroughBehavior": "when_no_match",

```



```
    "httpMethod": "POST",
    "cacheNamespace": "roq9wj",
    "cacheKeyParameters": [
        "method.request.path.proxy"
    ],
    "type": "aws_proxy"
  }
}
```

Bei der Lambda-Proxy-Integration ordnet API Gateway zur Laufzeit eine eingehende Anfrage dem Eingabe-event-Parameter der Lambda-Funktion zu. Die Eingabe umfasst die Anforderungsmethode, den Pfad, Header, jegliche Zeichenparameter, jegliche Nutzlast, zugehörigen Kontext und alle definierten Stufenvariablen. Eine Beschreibung des Eingabeformats finden Sie unter [Eingabeformat einer Lambda-Funktion für die Proxy-Integration](#). Damit API Gateway die Lambda-Ausgabe erfolgreich den HTTP-API-Antworten zuordnen kann, muss die Lambda-Funktion das Ergebnis in dem unter [Ausgabeformat einer Lambda-Funktion für die Proxy-Integration](#) beschriebenen Format ausgeben.

Bei der Lambda-Proxy-Integration einer Proxy-Ressource durch die ANY-Methode dient die einzelne Backend-Lambda-Funktion als Ereignis-Handler für alle Anfragen über die Proxy-Ressource. Beispielsweise können Sie zum Protokollieren der Datenverkehrsmuster ein Mobilgerät veranlassen, seinen Standort mit Angaben zu Bundesland, Stadt, Straße und Gebäude zu senden, indem Sie eine Anforderung mit `/state/city/street/house` im URL-Pfad für die Proxy-Ressource übermitteln. Die Backend-Lambda-Funktion kann dann den URL-Pfad analysieren und die Location-tuples in eine DynamoDB-Tabelle einfügen.

Benutzerdefinierte Lambda-Integrationen in API Gateway einrichten

Um zu zeigen, wie die benutzerdefinierte (nicht-Proxy-)Lambda-Integration eingerichtet wird, erstellen wir ein API in API Gateway, um die GET `/greeting?greeter={name}`-Methode zum Aufrufen einer Lambda-Funktion bereitzustellen. Verwenden Sie eine der folgenden Lambda-Beispielfunktionen für Ihre API.

Verwenden Sie eine der folgenden Lambda-Beispielfunktionen:

Node.js

```
'use strict';
var days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
  'Saturday'];
var times = ['morning', 'afternoon', 'evening', 'night', 'day'];

export const handler = async(event) => {
  console.log(event);
  // Parse the input for the name, city, time and day property values
  let name = event.name === null || event.name === undefined || event.name === "" ?
'you' : event.name;
  let city = event.city === undefined ? 'World' : event.city;
  let time = times.indexOf(event.time)<0 ? 'day' : event.time;
  let day = days.indexOf(event.day)<0 ? null : event.day;

  // Generate a greeting
  let greeting = 'Good ' + time + ', ' + name + ' of ' + city + '. ';
  if (day) greeting += 'Happy ' + day + '!';

  // Log the greeting to CloudWatch
  console.log('Hello: ', greeting);

  // Return a greeting to the caller
  return greeting;
};
```

Python

```
import json

def lambda_handler(event, context):
    print(event)
    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
        res['body'] = "Hello, World"
```

```
elif (event['greeter']):
    res['body'] = "Hello, " + event['greeter'] + "!"
else:
    raise Exception('Missing the required greeter parameter.')

return res
```

Die Funktion gibt die Nachricht "Hello, {name}!" zurück, sofern der greeter-Parameterwert keine leere Zeichenfolge ist. Wenn der "Hello, World!"-Wert eine leere Zeichenfolge ist, wird als Antwort greeter zurückgegeben. Die Funktion gibt eine Fehlermeldung zurück, "Missing the required greeter parameter." wenn der "greeter"-Parameter in der eingehenden Anforderung nicht festgelegt ist. Wir geben der Funktion den Namen HelloWorld.

Sie können sie in der Lambda-Konsole oder mithilfe der AWS CLI erstellen. In diesem Abschnitt definieren wir diese Funktion mithilfe des folgenden ARN:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Wenn die Lambda-Funktion im Backend eingerichtet ist, fahren Sie mit dem Einrichten der API fort.

So richten Sie die benutzerdefinierte Lambda-Integration mit dem AWS CLI

1. Verwenden Sie den folgenden [create-rest-api](#) Befehl, um eine API zu erstellen:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "name": "HelloWorld (AWS CLI)",
  "id": "te6si5ach7",
  "rootResourceId" : "krznpq9xpg",
  "createdDate": 1508461860
}
```

In diesem Beispiel verwenden Sie durchgängig die API id (te6si5ach7) und die rootResourceId (krznpq9xpg).

2. Verwenden Sie den folgenden [create-rest-api](#)-Befehl, um eine API-Gateway-[Ressource](#) mit dem Namen /greeting zu erstellen:

```
aws apigateway create-resource \  
  --rest-api-id te6si5ach7 \  
  --parent-id krznpq9xpg \  
  --path-part greeting
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "path": "/greeting",  
  "pathPart": "greeting",  
  "id": "2jf6xt",  
  "parentId": "krznpq9xpg"  
}
```

Sie verwenden den `id`-Wert der Ressource `greeting` (`2jf6xt`), um im nächsten Schritt eine Methode für die Ressource `/greeting` zu erstellen.

3. Verwenden Sie den folgenden [put-method](#)-Befehl, um eine API-Methodenanforderung von `GET /greeting?greeter={name}` zu erstellen:

```
aws apigateway put-method --rest-api-id te6si5ach7 \  
  --resource-id 2jf6xt \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters method.request.querystring.greeter=false
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "apiKeyRequired": false,  
  "httpMethod": "GET",  
  "authorizationType": "NONE",  
  "requestParameters": {  
    "method.request.querystring.greeter": false  
  }  
}
```

Diese API-Methode gestattet dem Client, einen Gruß über die Lambda-Funktion am Backend zu empfangen. Der `greeter`-Parameter ist optional, da das Backend entweder einen anonymen Aufrufer oder einen selbst identifizierten Aufrufer verarbeiten sollte.

4. Verwenden Sie den folgenden [put-method-response](#)-Befehl, um die 200 OK Antwort auf die Methodenanforderung von einzurichten `GET /greeting?greeter={name}`:

```
aws apigateway put-method-response \  
  --rest-api-id te6si5ach7 \  
  --resource-id 2jf6xt \  
  --http-method GET \  
  --status-code 200
```

5. Verwenden Sie den folgenden [put-integration](#)-Befehl, um die Integration der `GET /greeting?greeter={name}`-Methode mit einer Lambda-Funktion mit dem Namen `HelloWorld` einzurichten. Die Funktion beantwortet die Anforderung mit der Nachricht `"Hello, {name}!"`, sofern der `greeter`-Parameter angegeben wurde, oder `"Hello, World!"`, wenn kein Abfragezeichenfolgeparameter festgelegt wurde.

```
aws apigateway put-integration \  
  --rest-api-id te6si5ach7 \  
  --resource-id 2jf6xt \  
  --http-method GET \  
  --type AWS \  
  --integration-http-method POST \  
  --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \  
  --request-templates '{"application/json":{"\greeter\  
\"$input.params('greeter')\"}}' \  
  --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Die hier bereitgestellte Mapping-Vorlage übersetzt den `greeter`-Abfragezeichenfolgeparameter in den `greeter`-Parameter der JSON-Nutzlast. Dies ist notwendig, da die Eingabe einer Lambda-Funktion im Textkörper ausgedrückt werden muss.

Important

Für Lambda-Integrationen müssen Sie entsprechend der [Spezifikation der Lambda-Service-Aktion für Funktionsaufrufe](#) die HTTP-Methode `POST` für die Integrationsanfrage verwenden. Der `uri`-Parameter ist der ARN der Aktion, die die Funktion aufruft. Die Ausgabe sieht wie folgt aus:

```
{
  "passthroughBehavior": "WHEN_NO_MATCH",
  "cacheKeyParameters": [],
  "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
  "httpMethod": "POST",
  "requestTemplates": {
    "application/json": "{\"greeter\": \"${input.params('greeter')}\"}"
  },
  "cacheNamespace": "krznpq9xpg",
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "type": "AWS"
}
```

Die IAM-Rolle `apigAwsProxyRole` muss über Richtlinien verfügen, die es dem `apigateway`-Service erlauben, Lambda-Funktionen aufzurufen. Anstatt eine IAM-Rolle für `credentials` bereitzustellen, können Sie den Befehl [add-permission](#) aufrufen, um ressourcenbasierte Berechtigungen hinzuzufügen. Auf diese Weise fügt die API Gateway-Konsole diese Berechtigungen hinzu.

- Verwenden Sie den folgenden [put-integration-response](#) Befehl, um die Integrationsantwort so einzurichten, dass die Ausgabe der Lambda-Funktion als `200 OK` Methodenantwort an den Client übergeben wird:

```
aws apigateway put-integration-response \
  --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method GET \
  --status-code 200 \
  --selection-pattern ""
```

Beim Festlegen des Auswahlmusters auf eine leere Zeichenfolge lautet die voreingestellte Antwort `200 OK`.

Die Ausgabe sieht wie folgt aus:

```
{
  "selectionPattern": "",
```

```
"statusCode": "200"
}
```

7. Verwenden Sie den folgenden [create-deployment](#)-Befehl, um die API für eine test-Stufe bereitzustellen:

```
aws apigateway create-deployment \  
  --rest-api-id te6si5ach7 \  
  --stage-name test
```

8. Testen Sie die API mithilfe des folgenden cURL-Befehls in einem Terminal:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?greeter=me' \  
  -H 'authorization: AWS4-HMAC-SHA256 Credential={access_key}/20171020/us-west-2/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date, Signature=f327...5751'
```

Asynchronen Aufruf der Backend-Lambda-Funktion einrichten

Bei der nicht-proxy (benutzerdefinierten) Lambda-Integration wird die Backend-Lambda-Funktion standardmäßig synchron aufgerufen. Dies ist das gewünschte Verhalten für die meisten REST-API-Operationen. Einige Anwendungen erfordern jedoch eine asynchrone Durchführung (als Batchvorgang oder Operation mit langer Latenzzeit), in der Regel durch eine separate Backend-Komponente. In diesem Fall wird die Lambda-Backend-Funktion asynchron aufgerufen, und die Frontend-REST-API-Methode gibt das Ergebnis nicht zurück.

Sie können die Lambda-Funktion für eine nicht-proxy Lambda-Integration so konfigurieren, dass sie asynchron aufgerufen wird, indem Sie 'Event' als [Lambda invocation type \(Lambda-Aufruftyp\)](#) festlegen. Gehen Sie hierzu folgendermaßen vor:

Asynchronen Lambda-Aufruf in der API Gateway-Konsole konfigurieren

Damit alle Aufrufe asynchron sind:

- Fügen Sie in Integrationsanforderung einen X-Amz-Invocation-Type-Header mit dem statischen Wert 'Event' hinzu.

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

1. Fügen Sie in Methodenanforderung einen `InvocationType`-Header hinzu.
2. Fügen Sie in Integrationsanforderung einen `X-Amz-Invocation-Type`-Header mit dem Mapping-Ausdruck `method.request.header.InvocationType` hinzu.
3. Clients können den `InvocationType: Event`-Header in API-Anforderungen für asynchrone Aufrufe oder `InvocationType: RequestResponse` für synchrone Aufrufe einschließen.

Asynchronen Lambda-Aufruf mit OpenAPI konfigurieren

Damit alle Aufrufe asynchron sind:

- Fügen Sie die `X-Amz-Invocation-Type` Kopfzeile zum `x-amazon-apigateway-integration` Abschnitt hinzu.

```
"x-amazon-apigateway-integration" : {
  "type" : "aws",
  "httpMethod" : "POST",
  "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
  "responses" : {
    "default" : {
      "statusCode" : "200"
    }
  },
  "requestParameters" : {
    "integration.request.header.X-Amz-Invocation-Type" : "'Event'"
  },
  "passthroughBehavior" : "when_no_match",
  "contentHandling" : "CONVERT_TO_TEXT"
}
```

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

1. Fügen Sie ein beliebiges [OpenAPI Path Item Object](#) dem folgenden Header hinzu.

```
"parameters" : [ {
  "name" : "InvocationType",
  "in" : "header",
  "schema" : {
    "type" : "string"
  }
}
```



```
} ]
```

2. Fügen Sie die X-Amz-Invocation-Type Kopfzeile zum x-amazon-apigateway-integrationAbschnitt hinzu.

```
"x-amazon-apigateway-integration" : {
    "type" : "aws",
    "httpMethod" : "POST",
    "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
    "responses" : {
        "default" : {
            "statusCode" : "200"
        }
    },
    "requestParameters" : {
        "integration.request.header.X-Amz-Invocation-Type" :
"method.request.header.InvocationType"
    },
    "passthroughBehavior" : "when_no_match",
    "contentHandling" : "CONVERT_TO_TEXT"
}
```

3. Clients können den InvocationType: Event-Header in API-Anforderungen für asynchrone Aufrufe oder InvocationType: RequestResponse für synchrone Aufrufe einschließen.

Konfigurieren Sie den asynchronen Lambda-Aufruf mit CloudFormation

Die folgenden CloudFormation Vorlagen zeigen, wie Sie den AWS::ApiGateway::Method für asynchrone Aufrufe konfigurieren.

Damit alle Aufrufe asynchron sind:

```
AsyncMethodGet:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref AsyncResource
    HttpMethod: GET
    ApiKeyRequired: false
    AuthorizationType: NONE
    Integration:
```

```

    Type: AWS
    RequestParameters:
      integration.request.header.X-Amz-Invocation-Type: "'Event'"
    IntegrationResponses:
      - StatusCode: '200'
    IntegrationHttpMethod: POST
    Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
    ${myfunction.Arn}$/invocations
    MethodResponses:
      - StatusCode: '200'

```

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

```

AsyncMethodGet:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref AsyncResource
    HttpMethod: GET
    ApiKeyRequired: false
    AuthorizationType: NONE
    RequestParameters:
      method.request.header.InvocationType: false
    Integration:
      Type: AWS
      RequestParameters:
        integration.request.header.X-Amz-Invocation-Type:
        method.request.header.InvocationType
      IntegrationResponses:
        - StatusCode: '200'
      IntegrationHttpMethod: POST
      Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
      ${myfunction.Arn}$/invocations
    MethodResponses:
      - StatusCode: '200'

```

Clients können den `InvocationType: Event-Header` in API-Anforderungen für asynchrone Aufrufe oder `InvocationType: RequestResponse` für synchrone Aufrufe einschließen.

Lambda-Fehler in API Gateway behandeln

Bei benutzerdefinierten Lambda-Integrationen müssen Sie Fehler, die von Lambda in der Integrationsantwort zurückgegeben werden, für Ihre Kunden zu Standard-HTTP-Fehlerantworten zuordnen. Andernfalls werden Lambda-Fehler standardmäßig als 200 OK-Meldungen zurückgegeben und ein solches Verhalten ist für die API-Benutzer nicht intuitiv.

Lambda kann zwei Fehlertypen zurückgeben, nämlich Standardfehler und benutzerdefinierte Fehler. In Ihrer API müssen Sie diese unterschiedlich handhaben.

Bei der Lambda-Proxy-Integration muss Lambda eine Ausgabe im folgenden Format zurückgeben:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}
```

Bei dieser Ausgabe entspricht `statusCode` einem 4XX-Client-Fehler bzw. einem 5XX-Server-Fehler. API Gateway behandelt diese Fehler über das Mapping des Lambda-Fehlers zu einer HTTP-Fehlerantwort gemäß den angegebenen `statusCode`. Damit API Gateway den Fehlertyp (z. B. `InvalidParameterException`) als Teil der Antwort an den Client weitergibt, muss die Lambda-Funktion einen Header (z. B. `"X-Amzn-ErrorType": "InvalidParameterException"`) in der `headers`-Eigenschaft enthalten.

Themen

- [Standard-Lambda-Fehler in API Gateway behandeln](#)
- [Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln](#)

Standard-Lambda-Fehler in API Gateway behandeln

Ein AWS Lambda Standardfehler hat das folgende Format:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
```

```
    ...  
  ]  
}
```

Hier ist `errorMessage` ein Zeichenfolgeausdruck des Fehlers. Der Wert `errorType` gibt eine sprachabhängige Fehler- oder Ausnahmeart an. `stackTrace` zeigt in einer Liste von Zeichenfolgeausdrücken das Stacktrace an, das zum Fehlerereignis geführt hat.

Betrachten Sie beispielsweise die folgende Lambda-Funktion JavaScript (Node.js).

```
export const handler = function(event, context, callback) {  
  callback(new Error("Malformed input ..."));  
};
```

Diese Funktion gibt den folgenden Lambda-Standardfehler mit der Fehlermeldung `Malformed input ...` zurück:

```
{  
  "errorMessage": "Malformed input ...",  
  "errorType": "Error",  
  "stackTrace": [  
    "export const handler (/var/task/index.js:3:14)"  
  ]  
}
```

Bei der folgenden Python-Lambda-Funktion wird beispielsweise eine `Exception` mit derselben Fehlermeldung `Malformed input ...` ausgegeben:

```
def lambda_handler(event, context):  
    raise Exception('Malformed input ...')
```

Diese Funktion gibt den folgenden Standard-Lambda-Fehler zurück:

```
{  
  "stackTrace": [  
    [  
      "/var/task/lambda_function.py",  
      3,  
      "lambda_handler",  
      "raise Exception('Malformed input ...')"
```

```
]
],
"errorType": "Exception",
"errorMessage": "Malformed input ..."
}
```

Beachten Sie, dass die Eigenschaftswerte `errorType` und `stackTrace` sprachabhängig sind. Der Standardfehler gilt auch für jedes Fehlerobjekt, das eine Erweiterung des `Error`-Objekts oder eine Subklasse der `Exception`-Klasse ist.

Um den Standard-Lambda-Fehler einer Methodenantwort zuzuordnen, müssen Sie zunächst einen HTTP-Statuscode für einen bestimmten Lambda-Fehler festlegen. Anschließend legen Sie ein reguläres Ausdrucksmuster für die [selectionPattern](#) Eigenschaft fest, die dem angegebenen HTTP-Statuscode [IntegrationResponse](#) zugeordnet ist. In der API-Gateway-Konsole wird dieses `selectionPattern` im Abschnitt Integrationsantwort unter jeder Integrationsantwort als Lambda-Fehler-Regex ausgewiesen.

 Note

API Gateway verwendet für das Antwort-Mapping Regexe-Anweisungen im Java-Pattern-Stil. Weitere Informationen finden Sie unter [Pattern](#) in der OracleDokumentation.

Verwenden Sie beispielsweise Folgendes, [put-integration-response](#) um einen neuen `selectionPattern` Ausdruck einzurichten:

```
aws apigateway put-integration-response --rest-api-id z0vprf0mdh --resource-id x3o5ih
--http-method GET --status-code 400 --selection-pattern "Malformed.*" --region us-
west-2
```

Stellen Sie sicher, dass Sie auch den entsprechenden Fehlercode (400) für die [Methodenantwort definieren](#). Andernfalls gibt API Gateway zur Laufzeit eine ungültige Konfigurationsfehler-Antwort aus.

 Note

Zur Laufzeit ordnet API Gateway die `errorMessage` des Lambda-Fehlers dem Muster des regulären Ausdrucks für die `selectionPattern`-Eigenschaft zu. Wenn es eine Zuordnung gibt, gibt API Gateway den Lambda-Fehler als HTTP-Antwort des entsprechenden HTTP-Statuscodes zurück. Liegt keine Übereinstimmung vor, gibt API Gateway den Fehler

als Standardmeldung zurück. Falls keine Standardmeldung konfiguriert ist, wird eine Fehlermeldung bezüglich einer ungültigen Konfiguration ausgegeben.

Wenn als `selectionPattern`-Wert für eine bestimmte Antwort `. *` festgelegt wird, wird diese Antwort wieder als Standardantwort festgelegt. Der Grund hierfür ist, dass ein solches Auswahlmuster allen Fehlermeldungen entspricht, einschließlich Null, d. h. nicht spezifizierten Fehlermeldungen. Das daraus resultierende Mapping überschreibt das des Standard-Mapping. Wenn Sie `. +` als Auswahlmuster zur Filterung von Antworten verwenden, beachten Sie, dass es eine Antwort mit einem Zeilenumbruch (`\n`) möglicherweise nicht erkennt.

Um einen vorhandenen `selectionPattern` Wert mithilfe von zu aktualisieren AWS CLI, rufen Sie die [update-integration-response](#) Operation auf, um den `/selectionPattern` Pfadwert durch den angegebenen Regex-Ausdruck des `Malformed*` Musters zu ersetzen.

Wenn Sie eine Integrationsantwort für einen bestimmten HTTP-Statuscode einrichten oder aktualisieren wollen, geben Sie den Ausdruck im Textfeld Lambda-Fehler-Regex ein, um den `selectionPattern`-Ausdruck über die Amazon API Gateway-Konsole zu definieren.

Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln

AWS Lambda Ermöglicht es Ihnen, anstelle des im vorherigen Abschnitt beschriebenen Standardfehlers ein benutzerdefiniertes Fehlerobjekt als JSON-Zeichenfolge zurückzugeben. Der Fehler kann ein beliebiges gültiges JSON-Objekt sein. Beispielsweise gibt die folgende Lambda-Funktion JavaScript (Node.js) einen benutzerdefinierten Fehler zurück:

```
export const handler = (event, context, callback) => {
  ...
  // Error caught here:
  var myErrorObj = {
    errorType : "InternalServerError",
    httpStatus : 500,
    requestId : context.awsRequestId,
    trace : {
      "function": "abc()",
      "line": 123,
      "file": "abc.js"
    }
  }
  callback(JSON.stringify(myErrorObj));
}
```

```
};
```

Das `myErrorObj`-Objekt muss in eine JSON-Zeichenfolge umgewandelt werden, bevor Sie `callback` zum Beenden der Funktion aufrufen. Andernfalls wird `myErrorObj` als `"[object Object]"`-Zeichenfolge zurückgegeben. Wenn eine Methode Ihrer API mit der vorhergehenden Lambda-Funktion integriert wird, erhält API Gateway eine Integrationsantwort mit dem folgenden Payload:

```
{
  "errorMessage": "{\\"errorType\\":\\"InternalServerError\\",\\"httpStatus\\":500,
\\"requestId\\":\\"e5849002-39a0-11e7-a419-5bb5807c9fb2\\",\\"trace\\":{\\"function\\":
\\"abc()\\",\\"line\\":123,\\"file\\":\\"abc.js\\"}}}"
}
```

Wie bei jeder Integrationsantwort können Sie diese Fehlermeldung unverändert an die Methodenantwort übergeben. Alternativ können Sie die Nutzlast mithilfe einer Zuordnungsvorlage in ein anderes Format umwandeln. Nehmen Sie zum Beispiel die folgende Text-Mapping-Vorlage für eine Methodenantwort mit dem Statuscode 500:

```
{
  errorMessage: $input.path('$.errorMessage');
}
```

Diese Vorlage wandelt den Integrationsantworttext mit der benutzerdefinierten JSON-Fehlerzeichenfolge in den folgenden Methodenantworttext um. Dieser Methodenantworttext enthält das benutzerdefinierte JSON-Fehlerobjekt:

```
{
  "errorMessage" : {
    errorType : "InternalServerError",
    httpStatus : 500,
    requestId : context.awsRequestId,
    trace : {
      "function": "abc()",
      "line": 123,
      "file": "abc.js"
    }
  }
};
```

Abhängig von Ihren API-Anforderungen müssen Sie möglicherweise einige oder alle benutzerdefinierte Fehlereigenschaften als Header-Parameter für die Methodenantwort übergeben. Dazu wenden Sie die benutzerdefinierten Fehler-Mappings aus dem Integrationsantworttext auf die Header der Methodenantwort an.

Beispielsweise definiert die folgende OpenAPI-Erweiterung das Mapping der Eigenschaften `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` und `errorMessage.trace` zu den Headern `error_type`, `error_status`, `error_trace_function` und `error_trace`:

```
"x-amazon-apigateway-integration": {
  "responses": {
    "default": {
      "statusCode": "200",
      "responseParameters": {
        "method.response.header.error_trace_function":
"integration.response.body.errorMessage.trace.function",
        "method.response.header.error_status":
"integration.response.body.errorMessage.httpStatus",
        "method.response.header.error_type":
"integration.response.body.errorMessage.errorType",
        "method.response.header.error_trace":
"integration.response.body.errorMessage.trace"
      },
      ...
    }
  }
}
```

Zur Laufzeit deserialisiert API Gateway den Parameter `integration.response.body` bei der Ausführung von Header-Mappings. Diese Deserialisierung gilt jedoch nur für body-to-header Zuordnungen für benutzerdefinierte Lambda-Fehlerantworten und nicht für Zuordnungen mit. body-to-body `$input.body`. Bei diesen custom-error-body-to-Header-Zuordnungen erhält der Client die folgenden Header als Teil der Methodenantwort, vorausgesetzt, dass die Header, und in der Methodenanforderung deklariert `error_status` sind. `error_trace` `error_trace_function` `error_type`

```
"error_status": "500",
"error_trace": "{ \"function\": \"abc()\", \"line\": 123, \"file\": \"abc.js\" }",
"error_trace_function": "abc()",
```



```
"error_type": "InternalServerError"
```

Die Eigenschaft `errorMessage.trace` des Integrationsantworttexts ist eine komplexe Eigenschaft. Sie wird dem Header `error_trace` als JSON-Zeichenfolge zugewiesen.

HTTP-Integrationen für REST APIs in API Gateway

Sie können eine API-Methode mit einem HTTP-Endpunkt mithilfe der HTTP-Proxy-Integration oder der benutzerdefinierten HTTP-Integration integrieren.

API Gateway unterstützt die folgenden Endpunkt-Ports: 80, 443 und 1024-65535.

Mit der Proxy-Integration ist die Einrichtung einfach. Sie müssen nur die HTTP-Methode und die HTTP-Endpunkt-URI entsprechend der Backend-Anforderungen festlegen, wenn Sie sich nicht mit der Codierung oder dem Caching von Inhalten befassen.

Mit der benutzerdefinierten Integration ist die Einrichtung komplexer. Zusätzlich zu den Einrichtungsschritten der Proxy-Integration müssen Sie angeben, wie die eingehenden Anforderungsdaten der Integrationsanforderung und die resultierenden Integrationsantwortdaten der Methodenantwort zugeordnet werden.

Themen

- [HTTP-Proxy-Integrationen in API Gateway einrichten](#)
- [Benutzerdefinierte HTTP-API-Integrationen in API Gateway einrichten](#)

HTTP-Proxy-Integrationen in API Gateway einrichten

Erstellen Sie zum Einrichten einer Proxy-Ressource mit dem HTTP-Proxy-Integrationstyp eine API-Ressource mit einem gierigen Pfadparameter (zum Beispiel `/parent/{proxy+}`) und integrieren Sie diese Ressource mit einem HTTP-Backend-Endpunkt (zum Beispiel `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) in die Methode ANY. Der gierige Pfadparameter muss am Ende des Ressourcenpfads stehen.

Wie bei einer Nicht-Proxy-Ressource können Sie eine Proxy-Ressource mit der HTTP-Proxy-Integration einrichten, indem Sie die API Gateway-Konsole verwenden, eine OpenAPI-Definitionsdatei importieren oder die API Gateway-REST-API direkt aufrufen. Detaillierte Anweisungen für die Verwendung der API Gateway-Konsole zur Konfiguration einer Proxy-Ressource mit der HTTP-API-Integration finden Sie unter [Tutorial: Erstellen einer REST-API mit HTTP-Proxy-Integration](#).

Die folgende OpenAPI-Definitionsdatei zeigt ein Beispiel für eine API mit einer Proxyressource, die in die [PetStore](#) Website integriert ist.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "2016-09-12T23:19:28Z",
    "title": "PetStoreWithProxyResource"
  },
  "paths": {
   ("/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.proxy": "method.request.path.proxy"
          },
          "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/{proxy}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "ANY",
          "cacheNamespace": "rbftud",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "http_proxy"
        }
      }
    }
  }
}
```

```

    }
  }
},
"servers": [
  {
    "url": "https://4z9giyi2c1.execute-api.us-east-1.amazonaws.com/{basePath}",
    "variables": {
      "basePath": {
        "default": "/test"
      }
    }
  }
]
}

```

OpenAPI 2.0

```

{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T23:19:28Z",
    "title": "PetStoreWithProxyResource"
  },
  "host": "4z9giyi2c1.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
   ("/{proxy+}"): {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ]
      }
    }
  }
}

```

```

    "responses": {},
    "x-amazon-apigateway-integration": {
      "responses": {
        "default": {
          "statusCode": "200"
        }
      },
      "requestParameters": {
        "integration.request.path.proxy": "method.request.path.proxy"
      },
      "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/{proxy}",
      "passthroughBehavior": "when_no_match",
      "httpMethod": "ANY",
      "cacheNamespace": "rbftud",
      "cacheKeyParameters": [
        "method.request.path.proxy"
      ],
      "type": "http_proxy"
    }
  }
}
}
}

```

In diesem Beispiel wird ein Cache-Schlüssel für den Pfadparameter `method.request.path.proxy` der Proxy-Ressource deklariert. Dies ist die Standardeinstellung, wenn Sie die API mit der API Gateway-Konsole erstellen. Der Basispfad der API (`/test` entspricht einer Phase) ist der PetStore Seite der Website (`/petstore`) zugeordnet. Die einzelne Integrationsanfrage spiegelt die gesamte PetStore Website wider und verwendet dabei die Greedy Path-Variable der API und die ANY Catch-All-Methode. Die folgenden Beispiele veranschaulichen diese Spiegelung.

- Festlegen von **ANY** als **GET** und **{proxy+}** als **pets**

Vom Frontend initiierte Methodenanforderung:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
```

Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die den ersten Batch Haustiere enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **GET** und **{proxy+}** als **pets?type=dog**

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
HTTP/1.1
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
```

Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die den ersten Batch der angegebenen Hunde enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **GET** und **{proxy+}** als **pets/{petId}**

Vom Frontend initiierte Methodenanforderung:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die das angegebene Haustier enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **POST** und **{proxy+}** als **pets**

Vom Frontend initiierte Methodenanforderung:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
```

```
Content-Length: ...
```

```
{
  "type" : "dog",
  "price" : 1001.00
}
```

An das Backend gesendete Integrationsanforderung:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
```

```
Content-Type: application/json
```

```
Content-Length: ...
```

```
{
  "type" : "dog",
  "price" : 1001.00
}
```

Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die das neu erstellte Haustier enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **GET** und **{proxy+}** als **pets/cat**

Vom Frontend initiierte Methodenanforderung:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

Die Laufzeit-Instance des Proxy-Ressourcenpfads entspricht keinem Backend-Endpunkt und die resultierende Anforderung ist ungültig. Dies hat zur Folge, dass eine Antwort 400 Bad Request mit der folgenden Fehlermeldung zurückgegeben wird.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    }
  ]
}
```

```
    },  
    {  
      "key": "Pet2.price",  
      "message": "Missing required field"  
    }  
  ]  
}
```

- Festlegen von **ANY** als **GET** und **{proxy+}** als **null**

Vom Frontend initiierte Methodenanforderung:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Die Zielressource ist der Proxy-Ressource übergeordnet, aber die Laufzeit-Instance der Methode ANY ist in der API auf dieser Ressource nicht definiert. Infolgedessen gibt diese GET-Anfrage eine 403 Forbidden-Antwort mit der Missing Authentication Token-Fehlermeldung zurück, die von API Gateway zurückgegeben wird. Wenn die API die Methode ANY oder GET in der übergeordneten Ressource (/) verfügbar macht, gibt der Aufruf eine Antwort 404 Not Found mit der Meldung Cannot GET /petstore zurück, wie vom Backend zurückgegeben.

Wenn die URL des Zielpunkts ungültig ist oder das HTTP-Verb gültig ist, aber nicht unterstützt wird, gibt das Backend für jede Clientanforderung die Antwort 404 Not Found zurück. Bei einer nicht unterstützten HTTP-Methode wird eine Antwort 403 Forbidden zurückgegeben.

Benutzerdefinierte HTTP-API-Integrationen in API Gateway einrichten

Mit der benutzerdefinierten HTTP-Integration, auch bekannt als Nicht-Proxy-Integration, haben Sie mehr Kontrolle darüber, welche Daten zwischen einer API-Methode und einer API-Integration übertragen werden sollen und wie Sie diese Daten übertragen. Dies können Sie mithilfe von Daten-Mappings tun.

Als Teil der Einrichtung der Methodenanforderung legen Sie die [requestParameters](#)-Eigenschaft für eine [Method](#)-Ressource fest. Dies gibt an, welche Methodenanforderungsparameter, die von dem Client bereitgestellt werden, den Integrationsanforderungsparametern oder den geltenden

Textkörpereigenschaften zugeordnet werden sollen, bevor sie an den Backend versendet werden. Anschließend legen Sie im Rahmen der Einrichtung der Integrationsanfrage die Eigenschaft [requestParameters](#) für die entsprechende [Integrationsressource](#) fest, um die parameter-to-parameter Zuordnungen anzugeben. Sie legen auch die [requestTemplates](#)-Eigenschaft fest, um die Mapping-Vorlagen anzugeben, eine für jeden unterstützten Inhaltstyp. Die Mapping-Vorlagen weisen Methodenanforderungsparameter oder Textkörper dem Anforderungstextkörper der Integration zu.

In ähnlicher Weise legen Sie im Rahmen der Einrichtung der Methodenantwort die Eigenschaft [responseParameters](#) für die Ressource fest. [MethodResponse](#) Dies gibt an, welche Methodenantwortparameter, die an den Client gesendet werden sollen, von den Integrationsantwortparametern oder bestimmten geltenden Textkörpereigenschaften zugeordnet werden sollen, die vom Backend zurückgegeben wurden. Anschließend richten Sie das [selectionPattern](#) ein, um eine Integrationsantwort basierend auf der Antwort des Backends auszuwählen. Bei einer Nicht-Proxy-HTTP-Integration handelt es sich dabei um einen regulären Ausdruck. Beispiel: Um alle 2xx-HTTP-Statuscodes von einem HTTP-Endpunkt dieser Ausgabebezuordnung zuzuweisen, verwenden Sie `2\d{2}`.

 Note

API Gateway verwendet für das Antwort-Mapping Regexe-Anweisungen im Java-Pattern-Stil. Weitere Informationen finden Sie unter [Pattern](#) in der OracleDokumentation.

Anschließend legen Sie im Rahmen der Einrichtung der Integrationsantwort die [ResponseParameters-Eigenschaft für](#) die entsprechende [IntegrationResponse](#)Ressource fest, um die Zuordnungen anzugeben. parameter-to-parameter Sie legen auch die [responseTemplates](#)-Zuordnung fest, um die Mapping-Vorlagen anzugeben, eine für jeden unterstützten Inhaltstyp. Die Mapping-Vorlagen weisen Methodenantwortparameter oder Textkörpereigenschaften der Integrationsantwort dem Antworttextkörper der Methode zu.

Weitere Hinweise zum Einrichten von Mapping-Vorlagen finden Sie unter [Datentransformationen für REST-APIs in API Gateway](#).

Streamen Sie die Integrationsantwort für Ihre Proxy-Integrationen in API Gateway

Sie können Ihre Proxyintegration so konfigurieren, dass sie die Art steuert, wie API Gateway die Integrationsantwort zurückgibt. Standardmäßig wartet API Gateway auf den Empfang der vollständigen Antwort, bevor die Übertragung gestartet wird. Wenn Sie jedoch den Antwortübertragungsmodus Ihrer Integration auf `STREAM` einstellen, wartet API Gateway nicht, bis eine Antwort vollständig berechnet ist, bevor sie an den Client gesendet wird. Das Antwort-Streaming funktioniert für alle Arten von REST-API-Endpunkttypen.

Verwenden Sie das Antwort-Streaming für die folgenden Anwendungsfälle:

- Senken Sie den `time-to-first-byte` (TTFB) für generative KI-Anwendungen wie Chatbots.
- Streamen Sie große Bild-, Video- oder Musikdateien, ohne eine vorsignierte S3-URL zu verwenden.
- Führen Sie lang andauernde Operationen durch und melden Sie inkrementelle Fortschritte, z. B. vom Server gesendete Ereignisse (SSE).
- Überschreiten Sie das Nutzlastlimit von 10 MB für Antworten von API Gateway.
- Überschreiten Sie das Timeout-Limit von 29 Sekunden von API Gateway, ohne eine Erhöhung des Integrations-Timeout-Limits zu beantragen.
- Empfangen Sie eine binäre Nutzlast, ohne die binären Medientypen zu konfigurieren.

Überlegungen zum Streaming von Antwort-Payloads

Die folgenden Überlegungen könnten sich auf Ihre Verwendung von Antwort-Payload-Streaming auswirken:

- Sie können das Streaming von Antwortnutzlasten nur für `HTTP_PROXY` und `AWS_PROXY` Integrationstypen verwenden. Dazu gehören Lambda-Proxy-Integrationen und private Integrationen, die Integrationen verwenden. `HTTP_PROXY`
- Die Standardeinstellung für den Übertragungsmodus ist `BUFFERED`. Um Antwortstreaming zu verwenden, müssen Sie den Antwortübertragungsmodus auf `STREAM` ändern.
- Antwortstreaming wird nur für REST unterstützt APIs.
- Anforderungsstreaming wird nicht unterstützt.
- Sie können Ihre Antwort bis zu 15 Minuten lang streamen.

- Bei Ihren Streams kann es zu Timeouts im Leerlauf kommen. Bei regionalen oder privaten Endpunkten beträgt das Timeout 5 Minuten. Bei Edge-optimierten Endpunkten beträgt das Timeout 30 Sekunden.
- Wenn Sie das Antwort-Streaming für eine regionale REST-API mit Ihrer eigenen CloudFront Distribution verwenden, können Sie ein Leerlaufzeitlimit von mehr als 30 Sekunden erreichen, indem Sie das Antwort-Timeout Ihrer Distribution erhöhen. CloudFront Weitere Informationen finden Sie unter [Antwort-Timeout](#).
- Wenn der Antwortübertragungsmodus auf eingestellt ist `STREAM`, kann API Gateway keine Funktionen unterstützen, die eine Pufferung der gesamten Integrationsantwort erfordern. Aus diesem Grund werden die folgenden Funktionen beim Antwort-Streaming nicht unterstützt:
 - Endpunkt-Caching
 - Kodierung von Inhalten. Wenn Sie Ihre Integrationsantwort komprimieren möchten, tun Sie dies in Ihrer Integration.
 - Transformation der Antwort mit VTL
- Innerhalb jeder Streaming-Antwort unterliegen die ersten 10 MB der Antwort-Nutzlast keinen Bandbreitenbeschränkungen. Antwort-Nutzdaten, die 10 MB überschreiten, sind auf 2 MB/s beschränkt.
- Wenn die Verbindung zwischen dem Client und API Gateway oder zwischen API Gateway und Lambda aufgrund eines Timeouts geschlossen wird, wird die Lambda-Funktion möglicherweise weiterhin ausgeführt. Weitere Informationen finden Sie unter [Lambda-Funktions-Timeout konfigurieren](#).
- Das Streaming von Antworten ist mit Kosten verbunden. Weitere Informationen finden Sie unter [API-Gateway-Preise](#).

Richten Sie eine HTTP-Proxyintegration mit Payload-Response-Streaming in API Gateway ein

Wenn Sie das Antwort-Payload-Streaming einrichten, geben Sie den Antwortübertragungsmodus in der Integrationsanforderung Ihrer Methode an. Sie konfigurieren diese Einstellungen in der Integrationsanfrage, um zu steuern, wie sich API Gateway vor und während der Integrationsantwort verhält. Wenn Sie Antwort-Streaming verwenden, können Sie das Integrations-Timeout auf bis zu 15 Minuten konfigurieren.

Wenn Sie Payload Response Streaming mit einer `HTTP_PROXY` Integration verwenden, sendet API Gateway den HTTP-Antwortstatuscode oder irgendwelche HTTP-Antwort-Header erst, wenn es alle Header vollständig empfangen hat.

Erstellen Sie eine HTTP-Proxy-Integration mit Payload-Antwort-Streaming

Das folgende Verfahren zeigt Ihnen, wie Sie eine neue API mit der `responseTransferMode` Einstellung auf `STREAM` importieren. Wenn Sie über eine bestehende Integrations-API verfügen und diese ändern möchten `responseTransferMode`, finden Sie weitere Informationen unter [the section called “Aktualisieren Sie den Antwortübertragungsmodus für eine HTTP-Proxyintegration”](#).

AWS-Managementkonsole

So erstellen Sie eine HTTP-Proxyintegration mit Payload-Response-Streaming

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie `Create Resource (Ressource erstellen)` aus.
4. Geben Sie für `Resource name (Ressourcenname)` **streaming** ein.
5. Wählen Sie `Create Resource (Ressource erstellen)` aus.
6. Wählen Sie die Ressource `/streaming` aus und wählen Sie `Methode erstellen` aus.
7. Wählen Sie als Methodentyp die Option `ANY` aus.
8. Wählen Sie unter `Integrationstyp` die Option `HTTP` aus.
9. Wählen Sie `HTTP-Proxyintegration`.
10. Wählen Sie für den Antwortübertragungsmodus `Stream` aus.
11. Wählen Sie für die HTTP-Methode eine Methode aus.
12. Geben Sie für `Endpoint-URL` einen Integrationsendpunkt ein. Stellen Sie sicher, dass Sie einen Endpunkt wählen, der eine große Nutzlast erzeugt, die zu Ihnen zurückgestreamt wird.
13. Wählen Sie `Methode erstellen` aus.

Nachdem Sie Ihre Methode erstellt haben, stellen Sie Ihre API bereit.

Stellen Sie Ihre API bereit

1. Klicken Sie auf `Deploy API`.
2. Wählen Sie für `Stufe` die Option `Neue Stufe` aus.
3. Geben Sie für `Stage name (Stufenname)` **prod** ein.
4. (Optional) Geben Sie unter `Description (Beschreibung)` eine Beschreibung ein.

5. Wählen Sie Bereitstellen.

AWS CLI

Um eine neue API mit Payload-Response-Streaming zu erstellen

1. Kopieren Sie die folgende Open API-Datei und speichern Sie sie dann unter. `ResponseStreamDemoSwagger.yaml` In dieser Datei `responseTransferMode` ist auf `eingestelltSTREAM`. Der Integrationsendpunkt ist auf `eingestellthttps://example.com`, wir empfehlen jedoch, ihn so zu ändern, dass er eine große Nutzlast erzeugt, die an Sie zurückgestreamt wird.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
paths:
  /streaming:
    get:
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "https://example.com"
        type: "http_proxy"
        timeoutInMillis: 900000
        responseTransferMode: "STREAM"
```

2. Verwenden Sie den folgenden `import-rest-api` Befehl, um Ihre OpenAPI-Definition zu importieren:

```
aws apigateway import-rest-api \
  --body 'fileb:///~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Verwenden Sie den folgenden `create-deployment` Befehl, um Ihre neue API in einer Phase bereitzustellen:

```
aws apigateway create-deployment \  
  --rest-api-id a1b2c3 \  
  --stage-name prod \  
  --region us-west-1
```

Aktualisieren Sie den Antwortübertragungsmodus für eine HTTP-Proxyintegration

Das folgende Verfahren zeigt, wie der Antwortübertragungsmodus für eine HTTP-Proxyintegration aktualisiert wird.

AWS-Managementkonsole

Um den Antwortübertragungsmodus für eine HTTP-Proxyintegration zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie eine Methode aus.
4. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
5. Wählen Sie für den Antwortübertragungsmodus Stream aus.
6. Wählen Sie Speichern.

Nachdem Sie Ihre Methode aktualisiert haben, stellen Sie Ihre API bereit.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

AWS CLI

Der folgende `update-integration` Befehl aktualisiert den Übertragungsmodus einer Integration von `BUFFERED` bis `STREAM`. Für alle vorhandenen APIs Integrationen ist der Antwortübertragungsmodus für alle Integrationen auf `BUFFERED` eingestellt.

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3 \  
  --resource-id aaa111 \  
  --http-method GET \  
  --patch-operations "op='replace',path='/responseTransferMode',value=STREAM" \  
  --region us-west-1
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden. Wenn Sie das Integrations-Timeout angepasst haben, wird dieser Timeout-Wert entfernt, da API Gateway Ihre Antwort bis zu 5 Minuten lang streamt.

Der folgende `update-integration` Befehl aktualisiert den Übertragungsmodus einer Integration von `STREAM` bis `BUFFERED`:

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3 \  
  --resource-id aaa111 \  
  --http-method GET \  
  --patch-operations "op='replace',path='/responseTransferMode',value=BUFFERED" \  
  --region us-west-1
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Richten Sie eine Lambda-Proxy-Integration mit Payload-Response-Streaming in API Gateway ein

Sie können die Antwort einer Lambda-Funktion streamen, um die TTFB-Leistung (Time to First Byte) zu verbessern, und Teilantworten an den Client zurücksenden, sobald sie verfügbar sind. API Gateway erfordert, dass Sie die [InvokeWithResponseStream](#) Lambda-API verwenden, um Ihre Lambda-Funktion aufzurufen. API Gateway übergibt ein Event-Objekt an die Lambda-Funktion. Die Backend-Lambda-Funktion analysiert die eingehenden Anfragedaten, um die eigene Antwort zu bestimmen. Damit API Gateway die Lambda-Ausgabe streamen kann, muss die Lambda-Funktion das von API Gateway geforderte [Format](#) ausgeben.

Unterschiede bei der Lambda-Proxyintegration zwischen Stream- und Buffered Response-Übertragungsmodus

In der folgenden Liste werden die Unterschiede zwischen einer Lambda-Proxyintegration und einer Lambda-Proxyintegration für Response-Streaming beschrieben:

- API Gateway verwendet die [InvokeWithResponseStream](#) API, um die Lambda-Proxyintegration für das Antwort-Streaming aufzurufen. Dies führt zu einer anderen URI, die wie folgt lautet:

```
arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-streaming-invocations
```

Dieser ARN verwendet ein anderes Datum für die API-Version und eine andere Serviceaktion als die Lambda-Proxyintegration.

Wenn Sie die API Gateway Gateway-Konsole für das Antwort-Streaming verwenden, verwendet die Konsole den richtigen URI für Sie.

- Bei einer Lambda-Proxyintegration sendet API Gateway die Antwort erst an den Client, nachdem er die vollständige Antwort von Lambda erhalten hat. In einer Lambda-Proxyintegration für Antwortstreaming startet API Gateway den Payload-Stream, nachdem es die gültigen Metadaten und das Trennzeichen von Lambda erhalten hat.
- Die Lambda-Proxyintegration für Response-Streaming verwendet dasselbe Eingabeformat wie die Proxyintegration, erfordert jedoch ein anderes Ausgabeformat.

Lambda-Proxy-Integrationsformat für Antwort-Streaming

Wenn API Gateway eine Lambda-Funktion mit Antwortstreaming aufruft, entspricht das Eingabeformat dem Eingabeformat einer Lambda-Funktion für die Proxyintegration. Weitere Informationen finden Sie unter [the section called “Eingabeformat einer Lambda-Funktion für die Proxy-Integration”](#).

Wenn Lambda eine Antwort an API Gateway streamt, muss die Antwort dem folgenden Format entsprechen. Dieses Format verwendet ein Trennzeichen, um die JSON-Metadaten von der unformatierten Nutzlast zu trennen. In diesem Fall werden die Nutzdaten so gestreamt, wie sie von Ihrer Streaming-Lambda-Funktion übertragen werden:

```
{
  "headers": {"headerName": "headerValue", ...},
```

```
"multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
"cookies" : ["cookie1", "cookie2"],
"statusCode": httpStatusCode
}<DELIMITER>PAYLOAD1 | PAYLOAD2 | PAYLOAD3
```

In der Konsolenausgabe:

- Die `statusCode` Schlüsselheaders, `multiValueHeaders`cookies, und können nicht spezifiziert werden, wenn keine zusätzlichen Antwortheader zurückgegeben werden sollen.
- Der `headers`Schlüssel kann nur einwertige Header enthalten
- Die Ausgabe erwartet, dass die Header entweder oder enthalten. `Transfer-Encoding: chunked` `Content-length: number` Wenn Ihre Funktion keinen dieser Header zurückgibt, hängt API Gateway ihn an den Antwortheader `Transfer-Encoding: chunked` an.
- Der `multiValueHeaders`Schlüssel kann mehrwertige Header sowie einwertige Header enthalten. Sie können den `multiValueHeaders`Schlüssel verwenden, um alle zusätzlichen Header anzugeben, einschließlich einzelner Werte.
- Wenn Sie Werte sowohl für `headers` als auch für `multiValueHeaders` angeben, führt API Gateway diese in einer einzigen Liste zusammen. Wenn in beiden das gleiche Schlüssel / Wert-Paar angegeben ist, werden nur die Werte von in der `multiValueHeaders`zusammengeführten Liste angezeigt.
- Bei den Metadaten muss es sich um ein gültiges JSON handeln.
Nur `headers`, `multiValueHeaders`, `cookies` und die `statusCode` Schlüssel werden unterstützt.
- Sie müssen nach dem Metadaten-JSON ein Trennzeichen angeben. Das Trennzeichen muss aus 8 Null-Bytes bestehen und es muss innerhalb der ersten 16 KB der Stream-Daten vorkommen.
- API Gateway benötigt kein bestimmtes Format für die Nutzlast der Methodenantwort.

Wenn Sie eine Funktions-URL verwenden, um Ihre Lambda-Funktion zu streamen, müssen Sie die Eingabe und die Ausgabe Ihrer Lambda-Funktion ändern, um diese Anforderungen zu erfüllen.

Wenn Ihre Lambda-Funktionsausgabe nicht den Anforderungen dieses Formats entspricht, ruft API Gateway möglicherweise trotzdem Ihre Lambda-Funktion auf. Die folgende Tabelle zeigt die Kombinationen aus API-Integrationsanforderungseinstellungen und Lambda-Funktionscode, die von API Gateway unterstützt werden. Dies beinhaltet die unterstützten Kombinationen für den Antwortübertragungsmodus von `buffered`.

Antwortübertragungsmodus	Der Funktionscode entspricht dem erforderlichen Format	Lambda-API aufrufen	Unterstützt von API Gateway
Streamen	Ja	InvokeWithResponseStream	Ja. API Gateway streamt Ihre Antwort.
Streamen	Nein	InvokeWithResponseStream	Nein. API Gateway ruft Ihre Lambda-Funktion auf und gibt eine 500-Fehlerantwort zurück.
Streamen	Ja	Invoke	Nein. API Gateway unterstützt diese Integrationskonfiguration nicht.
Streamen	Nein	Invoke	Nein. API Gateway unterstützt diese Integrationskonfiguration nicht.
Gepuffert	Ja	InvokeWithResponseStream	Nein. API Gateway unterstützt diese Integrationskonfiguration nicht.
Gepuffert	Nein	InvokeWithResponseStream	Nein. API Gateway unterstützt diese Integrationskonfiguration nicht.
Gepuffert	Ja	Invoke	API Gateway gibt die HTTP-Header und den Statuscode zurück, aber nicht den Antworttext.

Antwortübertragungsmodus	Der Funktionscode entspricht dem erforderlichen Format	Lambda-API aufrufen	Unterstützt von API Gateway
Gepuffert	Nein	Invoke	Ja. Dies ist eine Lambda-Proxy-Integration. Weitere Informationen finden Sie unter Lambda-Proxyintegration .

Konfigurieren Sie eine Lambda-Proxyintegration mit Payload-Response-Streaming in API Gateway

Wenn Sie das Antwort-Payload-Streaming einrichten, geben Sie den Übertragungsmodus in der Integrationsanforderung Ihrer Ressource an. Sie konfigurieren diese Einstellungen in der Integrationsanfrage, um zu steuern, wie sich API Gateway vor und während der Integrationsantwort verhält.

Beispiel für Lambda-Funktionen für Response-Streaming

Ihre Lambda-Funktion muss dem [the section called “Lambda-Proxy-Integrationsformat für Antwort-Streaming”](#) entsprechen. Wir empfehlen Ihnen, eine der drei Lambda-Beispielfunktionen zu verwenden, um das Antwortstreaming zu testen. Wenn Sie Ihre Lambda-Funktion erstellen, stellen Sie sicher, dass Sie Folgendes tun:

- Sorgen Sie für ein ausreichendes Timeout für Ihre Funktion. Wir empfehlen Ihnen, ein Timeout von mindestens 1 Minute zu konfigurieren, um mehr über das Antwort-Streaming zu erfahren. Wenn Sie Ihre Produktionsressourcen erstellen, stellen Sie sicher, dass Ihr Lambda-Funktions-Timeout den gesamten Anforderungszyklus abdeckt. Weitere Informationen finden Sie unter [Lambda-Funktions-Timeout konfigurieren](#).
- Verwenden Sie die neueste Runtime von Node.js.
- Verwenden Sie eine Region, in der Lambda-Antwort-Streaming verfügbar ist.

Using `HttpResponseStream.from`

Das folgende Codebeispiel streamt die JSON-Metadatenobjekte und Payloads mithilfe der Methode zurück zum Client, ohne die `aws.lambda.HttpResponseStream()` Pipeline-Methode

zu verwenden. Sie müssen das Trennzeichen nicht erstellen. Weitere Informationen finden Sie unter [Schreiben von Lambda-Funktionen, die für Antwortstreaming aktiviert sind](#).

```
export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
    const httpResponseMetadata = {
      "statusCode": 200,
      "headers": {
        "x-foo": "bar"
      },
      "multiValueHeaders": {
        "x-mv1": ["hello", "world"],
        "Set-Cookie": ["c1=blue", "c2=red"]
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
      httpResponseMetadata);
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("First payload ");
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("Final payload");
    responseStream.end();
  });
```

Using the pipeline method

Lambda empfiehlt, dass Sie beim Schreiben von Funktionen, die für Response-Streaming aktiviert sind, den `awslambda.streamifyResponse()` Decorator verwenden, den die systemeigenen Laufzeiten von Node.js bereitstellen, sowie die Methode `pipeline()`. Wenn Sie die Pipeline-Methode verwenden, müssen Sie das Trennzeichen nicht erstellen, Lambda erledigt das für Sie. Weitere Informationen finden Sie unter [Schreiben von Lambda-Funktionen, die für Antwortstreaming aktiviert sind](#).

Im folgenden Codebeispiel werden die JSON-Metadatenobjekte und drei Payloads zurück zum Client gestreamt.

```
import { pipeline } from 'node:stream/promises';
import { Readable } from 'node:stream';
```

```

export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
    const httpResponseMetadata = {
      statusCode: 200,
      headers: {
        "Content-Type": "text/plain",
        "X-Custom-Header": "Example-Custom-Header"
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
      httpResponseMetadata);

    const dataStream = Readable.from(async function* () {
      yield "FIRST payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "SECOND payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "THIRD payload\n";
      await new Promise(r => setTimeout(r, 1000));
    }());

    await pipeline(dataStream, responseStream);
  }
);

```

Without using the pipeline method

Das folgende Codebeispiel streamt die JSON-Metadatenobjekte und drei Payloads zurück zum Client, ohne die Methode zu verwenden. `awslambda.HttpResponseStream()` Ohne die `awslambda.HttpResponseStream()` Methode müssen Sie ein Trennzeichen von 8 Null-Bytes zwischen den Metadaten und der Payload einfügen.

```

export const handler = awslambda.streamifyResponse(async (event, response, ctx) => {
  response.write('{"statusCode": 200, "headers": {"hdr-x": "val-x"}}');
  response.write("\x00".repeat(8)); // DELIMITER
  await new Promise(r => setTimeout(r, 1000));

  response.write("FIRST payload");
  await new Promise(r => setTimeout(r, 1000));

  response.write("SECOND payload");
  await new Promise(r => setTimeout(r, 1000));
});

```

```
response.write("FINAL payload");
response.end();
});
```

Erstellen Sie eine Lambda-Proxy-Integration mit Payload-Response-Streaming

Das folgende Verfahren zeigt, wie Sie eine Lambda-Proxyintegration mit Payload-Response-Streaming erstellen. Verwenden Sie die Lambda-Beispielfunktion oder erstellen Sie Ihre eigene.

AWS-Managementkonsole

So erstellen Sie eine Lambda-Proxyintegration mit Payload-Response-Streaming

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Create Resource (Ressource erstellen) aus.
4. Geben Sie für Resource name (Ressourcenname) **streaming** ein.
5. Wählen Sie Create Resource (Ressource erstellen) aus.
6. Wählen Sie die Ressource /streaming aus und wählen Sie Methode erstellen aus.
7. Wählen Sie als Methodentyp die Option ANY aus.
8. Wählen Sie unter Integrationstyp die Option Lambda aus.
9. Wählen Sie Lambda-Proxyintegration.
10. Wählen Sie für den Antwortübertragungsmodus die Option Stream aus.
11. Wählen Sie für Lambda-Funktion den Namen Ihrer Lambda-Funktion.

Die API-Gateway-Konsole verwendet automatisch [InvokeWithResponseStream](#)API, um die Lambda-Funktion aufzurufen. Sie sind dafür verantwortlich, eine Lambda-Funktion mit Response-Streaming zu schreiben. Ein Beispiel finden Sie unter [the section called “Beispiel für Lambda-Funktionen für Response-Streaming”](#).

12. Wählen Sie Methode erstellen aus.

Nachdem Sie Ihre Methode erstellt haben, stellen Sie Ihre API bereit.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

AWS CLI

Das folgende Verfahren zeigt Ihnen, wie Sie eine neue API mit der `responseTransferMode` Einstellung auf `importierenSTREAM`. Wenn Sie über eine bestehende Integrations-API verfügen und diese ändern möchten `responseTransferMode`, finden Sie weitere Informationen unter [the section called "Aktualisieren Sie den Antwortübertragungsmodus für eine Lambda-Proxyintegration"](#).

So erstellen Sie eine neue API mit Payload-Response-Streaming

1. Kopieren Sie die folgende Open API-Datei und speichern Sie sie dann unter. `ResponseStreamDemoSwagger.yaml` In dieser Datei `responseTransferMode` ist auf `eingestelltSTREAM`, und der Integrations-URI ist auf `eingestelltarn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-streaming-invocations`.

Ersetzen Sie den Funktionsnamen von `my-function` durch eine Streaming-fähige Funktion und ersetzen Sie die Anmeldeinformationen durch eine IAM-Rolle, deren Richtlinien es dem `apigateway` Dienst ermöglichen, Lambda-Funktionen aufzurufen.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
paths:
```

```
/lambda:
  get:
    x-amazon-apigateway-integration:
      httpMethod: "POST"
      uri: "arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-
streaming-invocations"
      type: "aws_proxy"
      timeoutInMillis: 90000
      responseTransferMode: "STREAM"
      credentials: "arn:aws:iam::111122223333:role/apigateway-lambda-role"
```

Anstatt eine IAM-Rolle für Anmeldeinformationen bereitzustellen, können Sie den `add-permission` Befehl für Lambda verwenden, um ressourcenbasierte Berechtigungen hinzuzufügen.

2. Verwenden Sie den folgenden `import-rest-api` Befehl, um Ihre OpenAPI-Definition zu importieren:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Verwenden Sie den folgenden `create-deployment` Befehl, um Ihre neue API in einer Phase bereitzustellen:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c2 \
  --stage-name prod \
  --region us-west-1
```

Aktualisieren Sie den Antwortübertragungsmodus für eine Lambda-Proxyintegration

Das folgende Verfahren zeigt, wie der Antwortübertragungsmodus für eine Lambda-Proxyintegration aktualisiert wird. Wenn Sie den Antwortübertragungsmodus auf Streaming ändern, aktualisieren Sie Ihre Lambda-Funktion, sodass sie den Anforderungen für das Antwortstreaming entspricht. Verwenden Sie die Lambda-Beispielfunktion oder erstellen Sie Ihre eigene.

AWS-Managementkonsole

So aktualisieren Sie den Antwortübertragungsmodus für eine Lambda-Proxyintegration

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie eine Methode aus.
4. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
5. Wählen Sie für den Antwortübertragungsmodus Stream aus.
6. Wählen Sie für Lambda-Funktion den Namen Ihrer Lambda-Funktion.
7. Wählen Sie Speichern.

Nachdem Sie Ihre Methode aktualisiert haben, stellen Sie Ihre API bereit.

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **prod** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

AWS CLI

1. Aktualisieren Sie Ihre Lambda-Funktion so, dass sie streaming-fähig ist.
2. Verwenden Sie den folgenden AWS CLI Befehl, um den Integrations-URI und den Antwortübertragungsmodus Ihrer Integration zu aktualisieren:

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3 \  
  --resource-id aaa111 \  
  --http-method ANY \  
  --patch-operations "[{\"op\":\"replace\",\"path\":\"/uri\",\"value  
  \":\"arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
```



```
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-  
streaming-invocations\"}, {\"op\": \"replace\", \"path\": \"/responseTransferMode  
\", \"value\": \"STREAM\"}]\" \\  
--region us-west-1
```

3. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Probleme mit dem Antwort-Streaming in API Gateway beheben

Die folgenden Anleitungen zur Fehlerbehebung können Ihnen helfen, Probleme mit Ihren Benutzern zu lösen APIs , die Antwort-Streaming verwenden.

Allgemeine Problembehebung

Sie können den Test-Tab [TestInvokeMethod](#) oder den Test-Tab der Konsole verwenden, um Ihre Stream-Antwort zu testen. Die folgenden Überlegungen könnten sich auf Ihre Verwendung von Test Invoke für das Antwort-Streaming auswirken:

- Wenn Sie Ihre Methode testen, puffert API Gateway Ihre gestreamte Antwort-Payload. Sobald eine der folgenden Bedingungen erfüllt ist, gibt API Gateway eine einmalige Antwort zurück, die die gepufferte Nutzlast enthält:
 - Die Anfrage ist abgeschlossen
 - 35 Sekunden sind vergangen
 - Mehr als 1 MB Antwort-Nutzdaten wurden gepuffert
- Wenn mehr als 35 Sekunden vergehen, bis Ihre Methode einen HTTP-Antwortstatus und alle Header zurückgibt, ist der zurückgegebene Antwortstatus 0. TestInvokeMethod
- API Gateway erstellt keine Ausführungsprotokolle.

Nachdem Sie Ihre API bereitgestellt haben, können Sie Ihre Stream-Antwort mit einem curl-Befehl testen. Wir empfehlen, dass Sie die `-i` Option verwenden, um Protokollantwort-Header in die Ausgabe aufzunehmen. Verwenden Sie die Option `curl`, um die Antwortdaten so zu sehen, wie sie ankommen `--no-buffer`

Behebung von cURL-Fehlern

Wenn Sie eine Integration testen und der Fehler angezeigt wird, stellen Sie sicher `curl: (18) transfer closed with outstanding read data remaining`, dass das Timeout Ihrer Integration lang genug ist. Wenn Sie eine Lambda-Funktion verwenden, müssen Sie das Antwort-

Timeout der Lambda-Funktion aktualisieren. Weitere Informationen finden Sie unter [Lambda-Funktions-Timeout konfigurieren](#).

Problembehandlung mithilfe der Zugriffsprotokollierung

Sie können Zugriffsprotokolle für Ihre REST-API-Phase verwenden, um Ihren Antwortstream zu protokollieren und Fehler zu beheben. Zusätzlich zu den vorhandenen Variablen können Sie die folgenden Zugriffsprotokollvariablen verwenden:

`$context.integration.responseTransferMode`

Der Antwortübertragungsmodus Ihrer Integration. Dies kann entweder BUFFERED oder STREAMED sein.

`$context.integration.timeToAllHeaders`

Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway die Integrationsverbindung herstellt, und dem Empfang aller Integrationsantwort-Header vom Client.

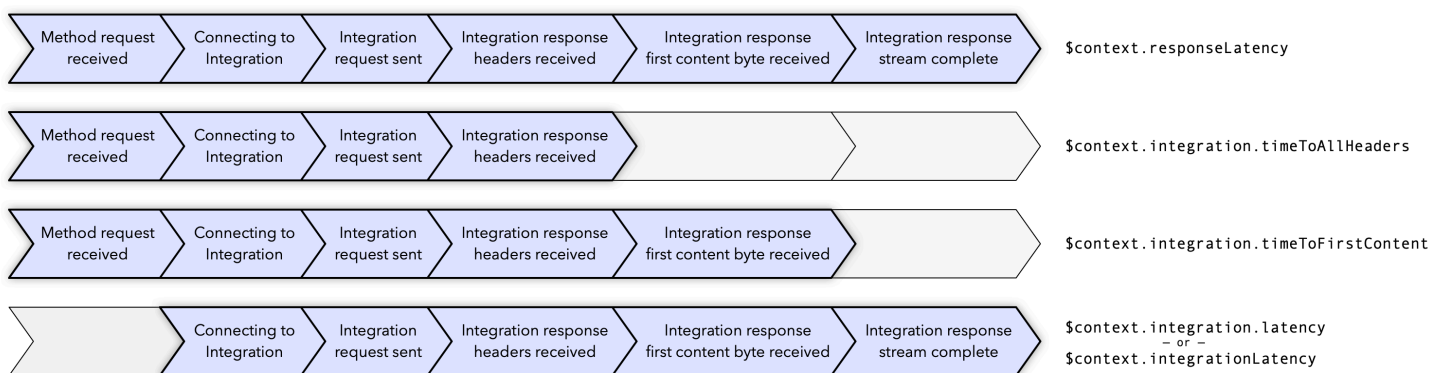
`$context.integration.timeToFirstContent`

Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway die Integrationsverbindung herstellt, und dem Empfang der ersten Inhaltsbytes.

`$context.integration.latency` oder `$context.integrationLatency`

Der Zeitpunkt, zu dem API Gateway die Integrationsverbindung herstellt, bis der Integrationsantwortstream abgeschlossen ist.

Die folgende Abbildung zeigt, wie diese Zugriffsprotokollvariablen verschiedene Komponenten eines Antwortstreams darstellen.



Weitere Informationen zu Zugriffsprotokollen finden Sie unter [the section called “CloudWatch protokolliert”](#). Sie können X-Ray auch verwenden, um Ihren Antwortstream zu überwachen. Weitere Informationen finden Sie unter [the section called “X-Ray”](#).

Private Integrationen für REST APIs in API Gateway

Verwenden Sie eine private Integration, um Ihre HTTP/HTTPS Ressourcen innerhalb einer Amazon-VPC für den Zugriff durch Kunden außerhalb der VPC verfügbar zu machen. Dadurch wird der Zugriff auf Ihre privaten VPC-Ressourcen über die VPC-Grenzen hinaus erweitert. Sie können den Zugriff auf Ihre API steuern, indem Sie eine der von API Gateway unterstützten [Autorisierungsmethoden](#) verwenden.

Um eine private Integration zu erstellen, müssen Sie zunächst einen VPC-Link erstellen. API Gateway unterstützt VPC-Links V2 für REST APIs. Mit VPC-Links V2 können Sie private Integrationen erstellen, die Ihre REST-API mit Application Load Balancern verbinden, ohne einen Network Load Balancer zu verwenden. Mit einem Application Load Balancer können Sie eine Verbindung zu ECSs Container-basierten Amazon-Anwendungen und vielen anderen Backends herstellen. VPC-Links V1 werden als älterer Integrationstyp betrachtet. Sie werden zwar von API Gateway unterstützt, wir empfehlen jedoch, keine neuen VPC-Links V1 zu erstellen.

Überlegungen

Die folgenden Überlegungen könnten sich auf Ihre Verwendung privater Integrationen auswirken:

- Alle Ressourcen müssen sich im Besitz derselben AWS-Konto Person befinden. Dazu gehören der Load Balancer, der VPC-Link und die REST-API.
- Standardmäßig verwendet der private Integrationsdatenverkehr das HTTP-Protokoll. Um HTTPS zu verwenden, geben Sie einen an [uri](#), der einen sicheren Servernamen enthält, z. B. `https://example.com:443/test`
- Bei einer privaten Integration schließt API Gateway den [Phase-Teil](#) des API-Endpunkts in die Anfrage an Ihre Backend-Ressourcen ein. Wenn Sie beispielsweise die `test` Phase einer API anfordern, nimmt API Gateway `test/path` in die Anfrage Ihrer private Integration auf. Um den Namen der Phase aus der Anfrage an Ihre Backend-Ressourcen zu entfernen, verwenden Sie die [Parameterzuordnung](#), um eine Überschreibung für die `$context.requestOverride.path` Variable zu erstellen.
- Private Integrationen mit AWS Cloud Map werden nicht unterstützt.

Themen

- [VPC-Links V2 im API Gateway einrichten](#)
- [Richten Sie eine private Integration ein](#)
- [Private Integration mithilfe von VPC-Links V1 \(veraltet\)](#)

VPC-Links V2 im API Gateway einrichten

VPC-Links ermöglichen es Ihnen, private Integrationen zu erstellen, die Ihre API-Routen mit privaten Ressourcen in einer VPC verbinden, z. B. Application Load Balancers oder Amazon ECS-Container-basierte Anwendungen. Eine private Integration verwendet einen VPC-Link V2, um Verbindungen zwischen API Gateway und gezielten VPC-Ressourcen zu kapseln. Sie können VPC-Links für verschiedene Ressourcen und APIs wiederverwenden.

Wenn Sie einen VPC-Link erstellen, erstellt und verwaltet API Gateway [elastische Netzwerkschnittstellen](#) für den VPC-Link V2 in Ihrem Konto. Dieser Vorgang kann einige Minuten dauern. Wenn ein VPC-Link V2 einsatzbereit ist, wechselt sein Status von PENDING zu AVAILABLE.

Note

Wenn 60 Tage lang kein Datenverkehr über den VPC-Link gesendet wird, wird er INACTIVE. Wenn sich ein VPC-Link in einem INACTIVE-Zustand befindet, werden alle Netzwerkschnittstellen des VPC-Links von API Gateway gelöscht. Dies führt dazu, dass API-Anforderungen, die von dem VPC-Link abhängen, fehlschlagen. Wenn API-Anforderungen wieder aufgenommen werden, stellt API Gateway Netzwerkschnittstellen wieder her. Es kann einige Minuten dauern, bis die Netzwerkschnittstellen erstellt und der VPC-Link reaktiviert wird. Sie können den VPC-Linkstatus verwenden, um den Status Ihres VPC-Links zu überwachen.

Erstellen Sie einen VPC-Link V2 mit dem AWS CLI

Um einen VPC-Link V2 zu erstellen, müssen alle beteiligten Ressourcen demselben AWS Konto gehören. Der folgende [create-vpc-link](#)-Befehl erstellt eine VPC-Verbindung:

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \  
  --subnet-ids subnet-aaaa subnet-bbbb \  
  --security-group-ids sg1234 sg5678
```

 Note

VPC-Links V2 sind unveränderlich. Nachdem Sie einen VPC-Link V2 erstellt haben, können Sie seine Subnetze oder Sicherheitsgruppen nicht mehr ändern.

Löschen Sie einen VPC-Link V2 mit dem AWS CLI

Der folgende [delete-vpc-link](#)-Befehl löscht eine VPC-Verbindung.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Verfügbarkeit nach Region

VPC-Links V2 werden in den folgenden Regionen und Availability Zones unterstützt:

Name der Region	Region	Unterstützte Availability Zones
USA Ost (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
USA Ost (Nord-Virginia)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
USA West (Nordkalifornien)	us-west-1	usw1-az1, usw1-az3
USA West (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asien-Pazifik (Hongkong)	ap-east-1	ape1-az2, ape1-az3
Asien-Pazifik (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3

Name der Region	Region	Unterstützte Availability Zones
Asia Pacific (Osaka)	ap-northeast-3	apne3-az1, apne3-az2, apne3-az3
Asien-Pazifik (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asien-Pazifik (Singapur)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3
Asien-Pazifik (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asien-Pazifik (Tokio)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Kanada (Zentral)	ca-central-1	cac1-az1, cac1-az2
Europa (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europa (Irland)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europa (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europa (Paris)	eu-west-3	euw3-az1, euw3-az3
Europa (Spain)	eu-south-2	eus2-az1, eus2-az2, eus2-az3
Europa (Stockholm)	eu-north-1	eun1-az1, eun1-az2, eun1-az3

Name der Region	Region	Unterstützte Availability Zones
Naher Osten (Bahrain)	me-south-1	mes1-az1, mes1-az2, mes1-az3
Südamerika (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Richten Sie eine private Integration ein

Um eine private Integration mit einem Application Load Balancer oder Network Load Balancer zu erstellen, erstellen Sie eine HTTP-Proxyintegration, geben den zu verwendenden [VPC-Link V2](#) an und geben den ARN eines Network Load Balancer oder eines Application Load Balancer an. Standardmäßig verwendet der private Integrationsdatenverkehr das HTTP-Protokoll. Um HTTPS zu verwenden, geben Sie einen an [uri](#), der einen sicheren Servernamen enthält, z. B. `https://example.com:443/test`. Ein vollständiges Tutorial zum Erstellen einer REST-API mit einer privaten Integration finden Sie unter [the section called “Tutorial: Erstellen einer REST-API mit einer privaten Integration”](#).

Erstellen einer privaten Integration

Das folgende Verfahren zeigt, wie Sie eine private Integration erstellen, die über einen VPC-Link V2 eine Verbindung zu einem Load Balancer herstellt.

AWS-Managementkonsole

Ein Tutorial zum Erstellen einer privaten Integration finden Sie unter [the section called “Tutorial: Erstellen einer REST-API mit einer privaten Integration”](#)

AWS CLI

Der folgende [put-integration-Befehl](#) erstellt eine private Integration, die über einen VPC-Link V2 eine Verbindung zu einem Load Balancer herstellt:

```
aws apigateway put-integration \
```

```
--rest-api-id abcdef123 \
--resource-id aaa000 \
--integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
--uri 'https://example.com:443/path' \
--http-method GET \
--type HTTP_PROXY \
--integration-http-method GET \
--connection-type VPC_LINK \
--connection-id bbb111
```

Anstatt die Verbindungs-ID direkt anzugeben, können Sie stattdessen eine Stufenvariable verwenden. Wenn Sie Ihre API in einer Phase bereitstellen, legen Sie die VPC-Link V2-ID fest. Der folgende [put-integration-Befehl](#) erstellt eine private Integration mithilfe einer Stage-Variablen für die VPC-Link V2-ID:

```
aws apigateway put-integration \
--rest-api-id abcdef123 \
--resource-id aaa000 \
--integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
--uri 'https://example.com:443/path' \
--http-method GET \
--type HTTP_PROXY \
--integration-http-method GET \
--connection-type VPC_LINK \
--connection-id "${stageVariables.vpcLinkId}"
```

Achten Sie darauf, den Ausdruck der Stufenvariablen (`${stageVariables.vpcLinkId}`) in doppelte Anführungszeichen zu setzen und das `$`-Zeichen zu maskieren.

OpenAPI

Sie können eine API mit der privaten Integration durch Importieren der API-OpenAPI-Datei einrichten. Die Einstellungen werden ähnlich wie die OpenAPI-Definitionen einer API mit HTTP-Integrationen festgelegt. Dabei gelten jedoch die folgenden Ausnahmen:

- Sie müssen `connectionType` explizit auf `VPC_LINK` festlegen.
- Sie müssen `connectionId` explizit auf die ID eines `VpcLinkIdV2` oder auf eine Stufenvariable festlegen, die auf die ID eines `VpcLinkIdV2` verweist.

- Der `uri` Parameter in der privaten Integration zeigt auf einen HTTP/HTTPS Endpunkt in der VPC, wird aber stattdessen verwendet, um den Host Header der Integrationsanfrage einzurichten.
- Der Parameter `uri` in der privaten Integration mit einem HTTPS-Endpunkt in der VPC wird verwendet, um den angegebenen Domännennamen anhand des Namens in dem auf dem VPC-Endpunkt installierten Zertifikats zu überprüfen.

Sie können eine Stufenvariable als Referenz auf die `VpcLinkIdV2`-ID verwenden. Sie können den ID-Wert auch `connectionId` direkt zuweisen.

Die folgende JSON-formatierte API-OpenAPI-Datei zeigt ein Beispiel für eine API mit einem VPC-Link, auf den eine Stufenvariable (`${stageVariables.vpcLinkIdV2}`) verweist:

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLinkV2"
  },
  "host": "abcdef123.execute-api.us-west-2.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "responses": {
          "default": {
```

```

        "statusCode": "200"
      }
    },
    "uri": "https://example.com:443/path",
    "passthroughBehavior": "when_no_match",
    "connectionType": "VPC_LINK",
    "connectionId": "${stageVariables.vpcLinkV2Id}",
    "integration-target": "arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011",
    "httpMethod": "GET",
    "type": "http_proxy"
  }
}
},
"definitions": {
  "Empty": {
    "type": "object",
    "title": "Empty Schema"
  }
}
}
}

```

Aktualisieren Sie eine private Integration

Das folgende Beispiel aktualisiert den VPC-Link V2 für eine private Integration.

AWS-Managementkonsole

Um eine private Integration zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API mit einer privaten Integration.
3. Wählen Sie die Ressource und Methode aus, die eine private Integration verwendet.
4. Klicken Sie in der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
5. Sie können die Einstellung Ihrer privaten Integration bearbeiten. Wenn Sie derzeit einen VPC-Link V1 verwenden, können Sie Ihren VPC-Link in einen VPC-Link V2 ändern.
6. Wählen Sie Speichern.

7. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

AWS CLI

Der folgende Befehl [update-integration](#) aktualisiert eine private Integration so, dass sie einen VPC-Link V2 verwendet:

```
aws apigateway update-integration \
  --rest-api-id a1b2c3d4e5 \
  --resource-id a1b2c3 \
  --http-method GET \
  --patch-operations "[{\\"op\\":\\"replace\\",\\"path\\":\\"/connectionId\\",\\"value\\":\\"pk0000\\"}, {\\"op\\":\\"replace\\",\\"path\\":\\"/uri\\",\\"value\\":\\"http://example.com\\"}, {\\"op\\":\\"replace\\",\\"path\\":\\"/integrationTarget\\",\\"value\\":\\"arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011\\"}]"
```

Private Integration mithilfe von VPC-Links V1 (veraltet)

Note

Die folgende Implementierung von privaten Integrationen verwendet VPC-Links V1. VPC-Links V1 sind Legacy-Ressourcen. Wir empfehlen die Verwendung von [VPC-Links V2 für REST APIs](#).

Um eine private Integration zu erstellen, müssen Sie zunächst einen Network Load Balancer erstellen. Ihr Network Load Balancer muss über einen [Listener](#) verfügen, der Anfragen an Ressourcen in Ihrer VPC weiterleitet. Zur Verbesserung der Verfügbarkeit Ihrer API stellen Sie sicher, dass Ihr Network Load Balancer den Datenverkehr an Ressourcen in mehr als einer Availability Zone im AWS-Region weiterleitet. Anschließend erstellen Sie einen VPC-Link, mit dem Sie Ihre API und Ihren Network Load Balancer verbinden. Nachdem Sie einen VPC-Link erstellt haben, erstellen Sie private Integrationen, um den Datenverkehr von Ihrer API über Ihre VPC-Verbindung und Ihren Network Load Balancer an Ressourcen in Ihrer VPC zu leiten. Der Network Load Balancer und die API müssen demselben AWS-Konto gehören.

Themen

- [Einen Network Load Balancer für private API-Gateway-Integrationen einrichten \(veraltet\)](#)

- [Erteilen Sie dem API Gateway Berechtigungen zum Erstellen eines VPC-Links \(Legacy\)](#)
- [Richten Sie eine API-Gateway-API mit privaten Integrationen mithilfe von AWS CLI \(Legacy\) ein](#)
- [API Gateway Gateway-Konten, die für private Integrationen verwendet werden \(veraltet\)](#)

Einen Network Load Balancer für private API-Gateway-Integrationen einrichten (veraltet)

 Note

Die folgende Implementierung von privaten Integrationen verwendet VPC-Links V1. VPC-Links V1 sind Legacy-Ressourcen. Wir empfehlen die Verwendung von [VPC-Links V2 für REST APIs](#).

Die folgende Prozedur umreißt die Schritte zur Einrichtung eines Network Load Balancers (NLB) für private API Gateway-Integrationen unter Verwendung der Amazon EC2-Konsole und bietet Referenzen für detaillierte Anweisungen zu jedem Schritt.

Für jede VPC, in der Sie Ressourcen haben, müssen Sie nur einen NLB und einen konfigurieren. VPCLink Der NLB unterstützt mehrere [Listener](#) und [Zielgruppen](#) pro NLB. Sie können jeden Dienst als einen bestimmten Listener auf der NLB konfigurieren und einen einzigen verwenden, um eine Verbindung VPCLink zur NLB herzustellen. Bei der Erstellung der privaten Integration in API Gateway definieren Sie jeden Service unter Verwendung des Ports, der dem Service zugewiesen ist. Weitere Informationen finden Sie unter [the section called “Tutorial: Erstellen einer REST-API mit einer privaten Integration”](#). Der Network Load Balancer und die API müssen demselben AWS-Konto gehören.

So erstellen Sie einen Network Load Balancer für die private Integration unter Verwendung der API Gateway-Konsole:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon EC2 EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Richten Sie einen Webserver in einer Amazon EC2-Instance ein. Ein Beispiel der Einrichtung finden Sie unter [Installieren eines LAMP-Webservers auf Amazon Linux 2](#).
3. Erstellen Sie einen Network Load Balancer, registrieren Sie die EC2-Instance für eine Zielgruppe und fügen Sie die Zielgruppe einem Listener des Network Load Balancer hinzu. Für weitere Informationen befolgen Sie die Anweisungen unter [Erste Schritte mit Network Load Balancern](#).
4. Nachdem der Network Load Balancer erstellt wurde, gehen Sie wie folgt vor:

- a. Notieren Sie sich den ARN des Network Load Balancers. Sie benötigen ihn, um einen VPC-Link in API Gateway für die Integration der API mit den VPC-Ressourcen hinter dem Network Load Balancer zu erstellen.
- b. Schalten Sie die Sicherheitsgruppenbewertung für PrivateLink aus.
 - Um die Sicherheitsgruppeneauswertung für den PrivateLink Datenverkehr mithilfe der Konsole zu deaktivieren, können Sie auf der Registerkarte Sicherheit auf Bearbeiten klicken. Deaktivieren Sie in den Sicherheitseinstellungen die Option Regeln für eingehenden PrivateLink Datenverkehr durchsetzen.
 - Verwenden Sie den folgenden [set-security-groups](#) Befehl, um die Sicherheitsgruppeneauswertung für den PrivateLink Datenverkehr zu deaktivieren:

```
aws elbv2 set-security-groups --load-balancer-  
arn arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/net/  
my-loadbalancer/abc12345 \  
--security-groups sg-123345a --enforce-security-group-inbound-rules-on-  
private-link-traffic off
```

Note

Fügen Sie keine Abhängigkeiten zu API Gateway hinzu, CIDRs da sie sich ohne vorherige Ankündigung ändern können.

Erteilen Sie dem API Gateway Berechtigungen zum Erstellen eines VPC-Links (Legacy)

Note

Die folgende Implementierung von privaten Integrationen verwendet VPC-Links V1. VPC-Links V1 sind Legacy-Ressourcen. Wir empfehlen die Verwendung von [VPC-Links V2 für REST APIs](#).

Damit Sie oder ein Benutzer in Ihrem Konto einen VPC-Link erstellen und pflegen können, müssen Sie bzw. der entsprechende Benutzer über Berechtigungen zum Erstellen, Löschen und Anzeigen der VPC-Endpoint-Servicekonfigurationen, zum Ändern der VPC-Endpoint-Service-Berechtigungen

und zum Prüfen der Load Balancer verfügen. Führen Sie die folgenden Schritte aus, um diese Berechtigungen zu erteilen.

So erteilen Sie Berechtigungen zum Erstellen, Aktualisieren und Löschen eines VPC-Links

1. Erstellen Sie eine IAM-Richtlinie wie etwa folgende:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:GET",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticloadbalancing:DescribeLoadBalancers"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpcEndpointServiceConfiguration",
        "ec2:DeleteVpcEndpointServiceConfigurations",
        "ec2:DescribeVpcEndpointServiceConfigurations",
        "ec2:ModifyVpcEndpointServicePermissions"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}  
]
```

Wenn Sie Markierungen für Ihren VPC-Link aktivieren möchten, müssen Sie Tagging-Operationen zulassen. Weitere Informationen finden Sie unter [the section called “Tagging-Operationen erlauben”](#).

2. Erstellen oder wählen Sie eine IAM-Rolle aus und fügen Sie die vorhergehende Richtlinie der Rolle an.
3. Weisen Sie die IAM-Rolle Ihnen oder einem Benutzer in Ihrem Konto zu, der VPC-Links erstellt.

Richten Sie eine API-Gateway-API mit privaten Integrationen mithilfe von AWS CLI (Legacy) ein

 Note

Die folgende Implementierung von privaten Integrationen verwendet VPC-Links V1. VPC-Links V1 sind Legacy-Ressourcen. Wir empfehlen die Verwendung von [VPC-Links V2 für REST APIs](#).

Das folgende Tutorial zeigt, wie Sie mit AWS CLI dem einen VPC-Link und eine private Integration erstellen. Die folgenden Voraussetzungen gelten:

- Sie benötigen einen Network Load Balancer, der mit Ihrer VPC-Quelle als Ziel erstellt und konfiguriert wurde. Weitere Informationen finden Sie unter [Einen Network Load Balancer für private API-Gateway-Integrationen einrichten \(veraltet\)](#). Dies muss mit Ihrer API AWS-Konto identisch sein. Sie benötigen den Network-Load-Balancer-ARN, um Ihren VPC-Link zu erstellen.
- Sie benötigen die Berechtigungen zum Erstellen eines VpcLink in Ihrer API, um einen VpcLink zu erstellen und zu verwalten. Zur Verwendung des VpcLink sind die Berechtigungen nicht erforderlich. Weitere Informationen finden Sie unter [Erteilen Sie dem API Gateway Berechtigungen zum Erstellen eines VPC-Links \(Legacy\)](#).

Um eine API mit der privaten Integration einzurichten, verwenden Sie AWS CLI

1. Verwenden Sie den folgenden [create-vpc-link](#) Befehl, um ein VpcLink Targeting für den angegebenen Network Load Balancer zu erstellen:

```
aws apigateway create-vpc-link \  
  --name my-test-vpc-link \  
  --target-arns arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/  
net/my-vpcLink-test-nlb/1234567890abcdef
```

Die Ausgabe dieses Befehls bestätigt den Eingang der Anfrage und zeigt den PENDING-Status des soeben erstellten VpcLink an.

```
{  
  "status": "PENDING",  
  "targetArns": [  
    "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-  
vpcLink-test-nlb/1234567890abcdef"  
  ],  
  "id": "gim7c3",  
  "name": "my-test-vpc-link"  
}
```

Es dauert 2-4 Minuten, bis API Gateway die Erstellung des VpcLink abgeschlossen hat. Wenn der Vorgang erfolgreich abgeschlossen wird, lautet der status AVAILABLE. Sie können dies mit dem folgenden [get-vpc-link](#) Befehl überprüfen:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Wenn der Vorgang fehlschlägt, erhalten Sie den Status FAILED und die statusMessage mit der Fehlermeldung. Wenn Sie beispielsweise versuchen, VpcLink mit einem Network Load Balancer zu erstellen, der bereits einem VPC-Endpoint zugewiesen ist, erhalten Sie Folgendes in der Eigenschaft statusMessage:

```
"NLB is already associated with another VPC Endpoint Service"
```

Erst nachdem der VpcLink erfolgreich erstellt wurde, können wir die API erstellen und über den VpcLink in die VPC-Ressource integrieren.

Notieren Sie sich den id-Wert der neu erstellten VpcLink. In dieser Beispielausgabe ist dies gim7c3. Sie benötigen ihn zum Einrichten der privaten Integration.

2. Verwenden Sie den folgenden [create-rest-api](#)-Befehl, um eine [RestApi](#)-API-Gateway-Ressource zu erstellen:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Notieren Sie den `id`-Wert von `RestApi` und den `rootResourceId`-Wert von `RestApi` in den zurückgegebenen Ergebnissen. Sie benötigen diesen Wert später, um weitere Vorgänge in der API, auszuführen.

Anschließend erstellen Sie eine API mit nur einer GET-Methode in der Root-Ressource (/) und integrieren die Methode im `VpcLink`.

3. Verwenden Sie den folgenden [put-method](#)-Befehl, um die GET /-Methode zu erstellen:

```
aws apigateway put-method \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --authorization-type "NONE"
```

Wenn Sie die Proxy-Integration mit dem `VpcLink` nicht verwenden, müssen Sie außerdem mindestens eine Methodenantwort des 200-Statuscodes einrichten. Sie verwenden die Proxy-Integration hier.

4. Nachdem Sie die GET /-Methode erstellt haben, richten Sie die Integration ein. Bei einer privaten Integration verwenden Sie den `connection-id`-Parameter zur Bereitstellung der `VpcLink`-ID. Sie können entweder eine Stufenvariable verwenden oder die `VpcLink`-ID direkt eingeben. Der Parameter `uri` wird nicht für das Routing von Anforderungen an Ihren Endpunkt verwendet, wohl aber zum Festlegen des Host-Headers und für die Zertifikatsvalidierung.

Use the VPC link ID

Führen Sie den [put-integration](#)-Befehl aus, um die `VpcLink`-ID direkt in der Integration zu verwenden:

```
aws apigateway put-integration \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \  
  \  
  --http-method GET \  
  --integration-type PRIVATE
```

```
--type HTTP_PROXY \  
--integration-http-method GET \  
--connection-type VPC_LINK \  
--connection-id gim7c3
```

Use a stage variable

Führen Sie den [put-integration](#)-Befehl aus, um eine Stufenvariable zu verwenden, die auf die VPC-Link-ID verweist. Wenn Sie Ihre API in einer Stufe bereitstellen, legen Sie die VPC-Link-ID fest.

```
aws apigateway put-integration \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \  
  \  
  --http-method GET \  
  --type HTTP_PROXY \  
  --integration-http-method GET \  
  --connection-type VPC_LINK \  
  --connection-id "${stageVariables.vpcLinkId}"
```

Stellen Sie sicher, dass Sie den Stufenvariablenausdruck in doppelte Anführungszeichen (`${stageVariables.vpcLinkId}`) setzen und das Zeichen `$` mit einem Escape-Zeichen versehen.

Sie können die Integration auch jederzeit aktualisieren, um die `connection-id` zu ändern. Verwenden Sie den [update-integration](#)-Befehl, um Ihre Integration zu aktualisieren:

```
aws apigateway update-integration \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --patch-operations '[{"op":"replace","path":"/  
connectionId","value":"${stageVariables.vpcLinkId}"}']'
```

Verwenden Sie unbedingt eine in Text umgewandelte JSON-Liste als `patch-operations`-Parameterwert.

Da Sie die private Proxy-Integration verwendet haben, ist Ihre API jetzt für die Bereitstellung und für Testläufe bereit.

5. Wenn Sie die Stufenvariable zur Definition Ihrer `connection-id` verwendet haben, müssen Sie Ihre API bereitstellen, um sie testen zu können. Führen Sie den [create-deployment](#)-Befehl aus, um Ihre API mit einer Stufenvariablen bereitzustellen:

```
aws apigateway create-deployment \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --variables vpcLinkId=gim7c3
```

Um die Stufenvariable mit einer anderen VpcLink-ID, beispielsweise *asf9d7*, zu aktualisieren, führen Sie den folgenden [update-stage](#)-Befehl aus:

```
aws apigateway update-stage \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Wenn Sie die `connection-id`-Eigenschaft mit dem VpcLink-ID-Literal hartcodieren, brauchen Sie Ihre API nicht bereitstellen, um sie testen zu können. Verwenden Sie den [test-invoke-method](#)-Befehl, um Ihre API zu testen, bevor sie bereitgestellt wird.

6. Rufen Sie Ihre API mit dem folgenden Befehl auf:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

Alternativ können Sie die invoke-URL Ihrer API in einem Webbrowser eingeben, um das Ergebnis anzuzeigen.

API Gateway Gateway-Konten, die für private Integrationen verwendet werden (veraltet)

Das folgende regionsspezifische API Gateway Gateway-Konto IDs wird Ihrem VPC-Endpunktdienst automatisch hinzugefügt, AllowedPrincipals wenn Sie ein erstellen. VpcLink

Region	Konto-ID
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833

Region	Konto-ID
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Scheinintegrationen für REST APIs in API Gateway

Amazon API Gateway unterstützt die Mock-Integration von API-Methoden. Diese Funktion ermöglicht es API-Entwicklern, API-Antworten direkt von API Gateway zu generieren, ohne ein Integrations-Backend zu benötigen. Als API-Developer können Sie diese Funktion nutzen, um abhängige Teams zu entsperren, die noch vor Beendigung der Projektentwicklung mit der API arbeiten müssen. Zudem lässt sich mit dieser Funktion eine Startseite für die API bereitstellen, die eine Übersicht über die API und die Navigation zu dieser bietet. Ein Beispiel für eine solche Startseite finden Sie in der Integrationsanforderung und -antwort der GET-Methode für die Stammressource der Beispiel-API unter [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Als API-Entwickler entscheiden Sie, wie API Gateway auf eine Mock-Integrationsanfrage antwortet. Dazu konfigurieren Sie die Integrationsanforderung und -antwort der Methode, um einer Antwort einen bestimmten Statuscode zuzuweisen. Damit eine Methode mit Mock-Integration eine 200-Antwort zurückgibt, konfigurieren Sie die Text-Mapping-Vorlage der Integrationsanforderung so, dass Folgendes zurückgegeben wird.

```
{"statusCode": 200}
```

Konfigurieren Sie eine 200-Integrationsantwort für folgende Text-Mapping-Vorlage, zum Beispiel:

```
{
  "statusCode": 200,
  "message": "Go ahead without me."
}
```

Damit die Methode beispielsweise eine 500-Fehlerantwort zurückgibt, richten Sie dem entsprechend die Text-Mapping-Vorlage der Integrationsanforderung so ein, dass Folgendes zurückgegeben wird.

```
{"statusCode": 500}
```

Richten Sie eine 500-Integrationsantwort beispielsweise mit der folgenden Mapping-Vorlage ein:

```
{  
  "statusCode": 500,  
  "message": "The invoked method is not supported on the API resource."  
}
```

Alternativ können Sie eine Methode der Mock-Integration die standardmäßige Integrationsantwort zurückgeben lassen, ohne die Mapping-Vorlage der Integrationsanforderung zu definieren. Die standardmäßige Integrationsantwort ist die mit einem nicht definierten HTTP status regex. Stellen Sie sicher, dass geeignete Pass-Through-Verhaltensweisen festgelegt wurden.

Note

Mock-Integrationen sind nicht für die Unterstützung großer Antwortvorlagen vorgesehen. Wenn Sie sie für Ihren Anwendungsfall benötigen, sollten Sie stattdessen die Verwendung einer Lambda-Integration in Betracht ziehen.

Mit der Mapping-Vorlage einer Integrationsanforderung können Sie Anwendungslogik einbringen, um basierend auf bestimmten Bedingungen zu entscheiden, welche Mock-Integrationsantwort zurückgegeben wird. Sie können bei der eingehenden Anforderung beispielsweise einen scope-Abfrageparameter verwenden, um zu bestimmen, ob eine Erfolgs- oder Fehlerantwort zurückgegeben wird:

```
{  
  #if( $input.params('scope') == "internal" )  
    "statusCode": 200  
  #else  
    "statusCode": 500  
  #end  
}
```

Auf diese Weise lässt die Methode der Mock-Integration interne Aufrufe durch und lehnt andere Aufruftypen mit einer Fehlermeldung ab.

In diesem Abschnitt wird beschrieben, wie die API Gateway-Konsole verwendet wird, um die Mock-Integration für eine API-Methode zu ermöglichen.

Themen

- [Mock-Integration mit der API Gateway-Konsole aktivieren](#)

Mock-Integration mit der API Gateway-Konsole aktivieren

Hierfür muss in API Gateway eine Methode verfügbar sein. Folgen Sie den Anweisungen in [Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen](#).

1. Wählen Sie eine API-Ressource aus und klicken Sie auf Methode erstellen.

Die Methode richten Sie wie folgt ein:

- a. Für Methodentyp wählen Sie eine HTTP-Methode aus.
 - b. Für den Integrationstyp wählen Sie Mock aus.
 - c. Wählen Sie Methode erstellen aus.
 - d. Klicken Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen auf Bearbeiten.
 - e. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen. Klicken Sie auf Abfragezeichenfolge hinzufügen und geben Sie **scope** als Namen ein. Dieser Abfrageparameter stellt fest, ob es sich um einen internen Aufrufer handelt.
 - f. Wählen Sie Speichern.
2. Klicken Sie auf der Registerkarte Methodenantwort auf Antwort erstellen und gehen Sie dann wie folgt vor:
 - a. Für HTTP-Status geben Sie **500** ein.
 - b. Wählen Sie Speichern.
 3. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
 4. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Add mapping template.

- b. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
- c. Geben Sie für Vorlagentext Folgendes ein:

```
{  
  #if( $input.params('scope') == "internal" )  
    "statusCode": 200  
  #else  
    "statusCode": 500  
  #end  
}
```

- d. Wählen Sie Speichern.
5. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
6. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:
- a. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
 - b. Geben Sie für Vorlagentext Folgendes ein:

```
{  
  "statusCode": 200,  
  "message": "Go ahead without me"  
}
```

- c. Wählen Sie Speichern.
7. Wählen Sie Create response (Antwort erstellen) aus.

Gehen Sie wie folgt vor, um eine Antwort 500 zu erstellen:

- a. Machen Sie für HTTP status regex (HTTP-Status-RegEx) den Eintrag **5\d{2}**.
- b. Wählen Sie für Status der Methodenantwort **500** aus.
- c. Wählen Sie Speichern.
- d. Klicken Sie unter 5\d{2} - Antwort auf Bearbeiten.
- e. Wählen Sie Zuordnungsvorlagen aus und klicken Sie dann auf Zuordnungsvorlage hinzufügen.
- f. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
- g. Geben Sie für Vorlagentext Folgendes ein:

```
{
```



```
"statusCode": 500,  
"message": "The invoked method is not supported on the API resource."  
}
```

- h. Wählen Sie Speichern.
- 8. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen. Gehen Sie wie folgt vor, um Ihre Mock-Integration zu testen:
 - a. Geben Sie unter Abfragezeichenfolgen `scope=internal` ein. Wählen Sie Test aus. Das Testergebnis zeigt Folgendes:

```
Request: /?scope=internal  
Status: 200  
Latency: 26 ms  
Response Body  
  
{  
  "statusCode": 200,  
  "message": "Go ahead without me"  
}  
  
Response Headers  
  
{"Content-Type":"application/json"}
```

- b. Geben Sie `scope=public` unter Query strings ein oder lassen Sie das Feld leer. Wählen Sie Test aus. Das Testergebnis zeigt Folgendes:

```
Request: /  
Status: 500  
Latency: 16 ms  
Response Body  
  
{  
  "statusCode": 500,  
  "message": "The invoked method is not supported on the API resource."  
}  
  
Response Headers
```

```
{"Content-Type":"application/json"}
```

Sie können auch Header in einer Mock-Integrationsantwort zurückgeben, indem Sie der Methodenantwort zunächst einen Header hinzufügen und anschließend ein Header-Mapping in der Integrationsantwort einrichten. Die API Gateway-Konsole setzt die CORS-Unterstützung über die Rückgabe von CORS-fähigen Headern um.

Validierung für REST APIs in API Gateway anfordern

Sie können API Gateway so konfigurieren, dass es eine grundlegende Validierung einer API-Anfrage durchführt, bevor es mit der Integrationsanfrage fortfährt. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage sofort fehl, gibt eine 400-Fehlerantwort an den Aufrufer zurück und veröffentlicht die Überprüfungsergebnisse in CloudWatch Logs. Damit werden unnötige Aufrufe im Backend vermieden. Vor allem aber können Sie die Validierung so speziell auf die Anwendung ausrichten. Sie können einen Anforderungstext validieren, indem Sie überprüfen, ob die erforderlichen Anforderungsparameter gültig sind und nicht den Wert Null haben, oder indem Sie für komplexere Datenvalidierungen ein Modellschema angeben.

Themen

- [Überblick über die grundlegende Anforderungsvalidierung in API Gateway](#)
- [Datenmodelle für REST APIs](#)
- [Grundlegende Anforderungsvalidierung in API Gateway einrichten](#)
- [AWS CloudFormation Vorlage einer Beispiel-API mit grundlegender Anforderungsvalidierung](#)

Überblick über die grundlegende Anforderungsvalidierung in API Gateway

API Gateway kann die grundlegende Anforderungsvalidierung durchführen, sodass Sie sich auf die anwendungsspezifische Validierung im Backend konzentrieren können. Für die grundlegende Validierung überprüft API Gateway eine oder beide der folgenden Bedingungen:

- Die erforderlichen Anforderungsparameter in der URI, der Abfragezeichenfolge und dem Header einer eingehenden Anforderung sind vorhanden und nicht leer. API Gateway überprüft nur das Vorhandensein eines Parameters und nicht den Typ oder das Format.

- Die entsprechenden Anforderungsnutzdaten entsprechen dem konfigurierten [JSON-Schema](#) der Methodenanforderung für einen bestimmten Inhaltstyp. Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungvalidierung nicht durchgeführt. Um dasselbe Modell unabhängig vom Inhaltstyp zu verwenden, legen Sie den Inhaltstyp für Ihr Datenmodell auf `$default` fest.

Zum Aktivieren der grundlegenden Validierung legen Sie Validierungsregeln für eine [Anforderungvalidierung](#) fest, fügen diese dem [Schema der Anforderungvalidierungen](#) für die API hinzu und ordnen anschließend die Validierung den einzelnen API-Methoden zu.

Note

Das Anfordern einer Textüberprüfung an und [Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway](#) sind zwei separate Themen. Wenn die Nutzlast einer Anforderung kein passendes Modellschema hat, können Sie Pass-Through wählen oder die ursprüngliche Nutzlast blockieren. Weitere Informationen finden Sie unter [Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway](#).

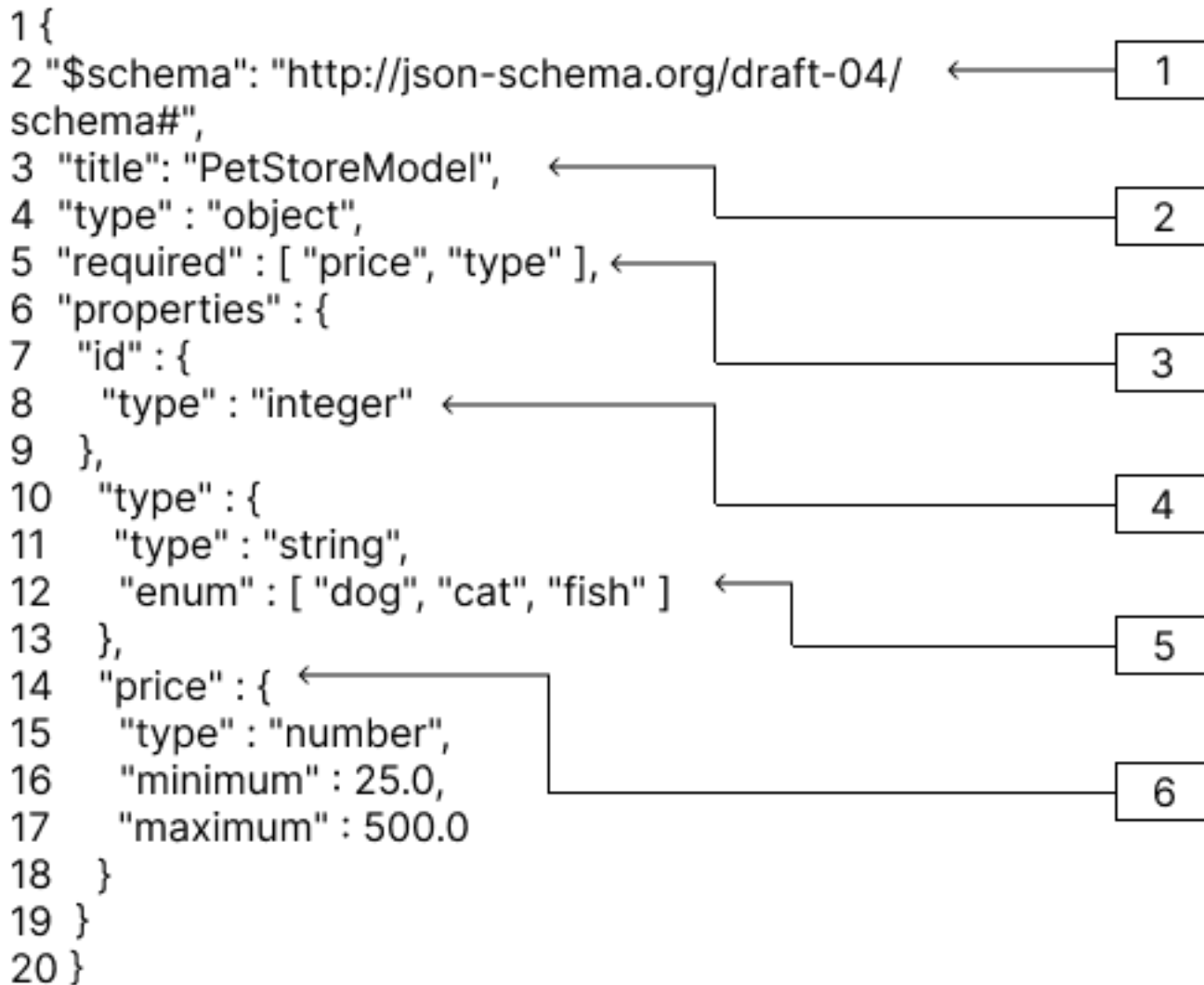
Datenmodelle für REST APIs

In API Gateway definiert ein Modell die Datenstruktur eines Payloads. In API Gateway werden Modelle unter Verwendung des [JSON-Schemaentwurfs 4](#) definiert. Das folgende JSON-Objekt enthält Beispieldaten im Tierhandlungsbeispiel.

```
{
  "id": 1,
  "type": "dog",
  "price": 249.99
}
```

Die Daten enthalten die `id`, den `type` und den `price` des Haustiers. Ein Modell dieser Daten ermöglicht Ihnen Folgendes:

- Verwenden der grundlegenden Anforderungvalidierung.
- Erstellen von Zuweisungsvorlagen für die Datentransformation.
- Erstellen eines benutzerdefinierten Datentyps (UDT), wenn Sie ein SDK generieren.



In diesem Modell gilt:

1. Das Objekt `$schema` stellt eine gültige JSON-Schema-Versionskennung dar. Dieses Schema ist der JSON-Schemaentwurf v4.
2. Das Objekt `title` ist eine lesbare Kennung des Modells. Dieser Titel ist `PetStoreModel`.
3. Das `required`-Validierungsschlüsselwort erfordert `type` und `price` für die grundlegende Anforderungvalidierung.
4. Die `properties` des Modells sind `id`, `type` und `price`. Jedes Objekt hat Eigenschaften, die im Modell beschrieben werden.
5. Das Objekt `type` kann nur die Werte `dog`, `cat` oder `fish` haben.

6. Das Objekt `price` ist eine Zahl und auf einen `minimum`-Wert von 25 und einen `maximum`-Wert von 500 beschränkt.

PetStore Modell

```
1 {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "title": "PetStoreModel",
4   "type" : "object",
5   "required" : [ "price", "type" ],
6   "properties" : {
7     "id" : {
8       "type" : "integer"
9     },
10    "type" : {
11      "type" : "string",
12      "enum" : [ "dog", "cat", "fish" ]
13    },
14    "price" : {
15      "type" : "number",
16      "minimum" : 25.0,
17      "maximum" : 500.0
18    }
19  }
20 }
```

In diesem Modell gilt:

1. In Zeile 2 stellt das Objekt `$schema` eine gültige JSON-Schema-Versionskennung dar. Dieses Schema ist der JSON-Schemaentwurf v4.
2. In Zeile 3 ist das Objekt `title` eine lesbare Kennung des Modells. Dieser Titel ist `PetStoreModel`.
3. In Zeile 5 erfordert das `required` Validierungsschlüsselwort `type` und `price` für die grundlegende Anforderungsvalidierung.
4. In den Zeilen 6 bis 17 sind die `properties` des Modells `id`, `type` und `price`. Jedes Objekt hat Eigenschaften, die im Modell beschrieben werden.
5. In Zeile 12 `type` kann das Objekt nur die Werte `dog`, `cat` oder `fish` haben.
6. In den Zeilen 14 bis 17 ist das Objekt `price` eine Zahl und durch einen `minimum`-Wert von 25 und einen `maximum`-Wert von 500 eingeschränkt.

Erstellen komplexerer Modelle

Sie können das `$ref`-Primitiv verwenden, um wiederverwendbare Definitionen für längere Modelle zu erstellen. Sie können beispielsweise eine Definition mit der Bezeichnung `Price` im `definitions`-Abschnitt zur Beschreibung des `price`-Objekts erstellen. Der Wert von `$ref` ist die `Price`-Definition.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
      "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
    "Price": {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Sie können auch auf ein anderes Modellschema verweisen, das in einer externen Modelldatei definiert ist. Stellen Sie den Wert der `$ref`-Eigenschaft auf die Position des Modells ein. Im folgenden Beispiel ist das `Price`-Modell im `PetStorePrice`-Modell in der API `a1234` definiert.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
```

```
"maximum": 500
}
```

Das längere Modell kann auf das PetStorePrice-Modell verweisen.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
      "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Verwendung von Ausgabedatenmodellen

Wenn Sie Ihre Daten umwandeln, können Sie in der Integrationsantwort ein Nutzlastmodell definieren. Ein Nutzlastmodell kann verwendet werden, wenn Sie ein SDK generieren. Bei stark typisierten Sprachen wie Java, Objective-C oder Swift entspricht das Objekt einem benutzerdefinierten Datentyp (UDT). API Gateway erstellt eine UDT, wenn Sie ihm bei der Generierung eines SDK ein Datenmodell zur Verfügung stellen. Weitere Informationen zu Datentransformationen finden Sie unter [Transformationen mit Zuweisungsvorlagen für REST-APIs in API Gateway](#).

Im folgenden Beispiel sehen Sie die Ausgabedaten einer Integrationsantwort.

```
{
  [
    {
      "description" : "Item 1 is a dog.",
      "askingPrice" : 249.99
    },
  ],
}
```

```
{
  "description" : "Item 2 is a cat.",
  "askingPrice" : 124.99
},
{
  "description" : "Item 3 is a fish.",
  "askingPrice" : 0.99
}
]
```

Im folgenden Beispiel sehen Sie ein Payload-Modell, das die Ausgabedaten beschreibt.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PetStoreOutputModel",
  "type" : "object",
  "required" : [ "description", "askingPrice" ],
  "properties" : {
    "description" : {
      "type" : "string"
    },
    "askingPrice" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Mit diesem Modell können Sie ein SDK aufrufen, um die Eigenschaftswerte `description` und `askingPrice` durch Lesen der Eigenschaften `PetStoreOutputModel[i].description` und `PetStoreOutputModel[i].askingPrice` zu erhalten. Wenn kein Modell zur Verfügung gestellt wird, verwendet API Gateway das leere Modell zur Erstellung einer Standard-UDT.

Nächste Schritte

- In diesem Abschnitt finden Sie Ressourcen, die Sie verwenden können, um mehr über die in diesem Thema vorgestellten Konzepte zu erfahren.

Sie können den Tutorials zur Validierung von Anfragen folgen:

- [Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole](#)
 - [Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI](#)
 - [Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition](#)
- Weitere Informationen zu Datentransformations- und Zuweisungsvorlagen finden Sie unter [Transformationen mit Zuweisungsvorlagen für REST-APIs in API Gateway](#).

Grundlegende Anforderungsvalidierung in API Gateway einrichten

In diesem Abschnitt wird gezeigt, wie die Anforderungsvalidierung für API Gateway mithilfe der Konsole und einer OpenAPI-Definition eingerichtet wird. AWS CLI

Themen

- [Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole](#)
- [Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI](#)
- [Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition](#)

Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole

Sie können die API-Gateway-Konsole verwenden, um eine Anfrage zu validieren, indem Sie einen von drei Validatoren für eine API-Anfrage auswählen:

- Text validieren.
- Abfragezeichenfolgeparameter und Header validieren.
- Text, Abfragezeichenfolgeparameter und Header validieren.

Wenn Sie einen der Validatoren auf eine API-Methode anwenden, fügt die API-Gateway-Konsole den Validator zur API-Map hinzu. [RequestValidators](#)

Um diesem Tutorial zu folgen, verwenden Sie eine CloudFormation Vorlage, um eine unvollständige API Gateway zu erstellen. Diese API hat eine `/validator`-Ressource mit den Methoden GET und POST. Beide Methoden sind in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets-HTTP-Endpoint` integriert. Sie werden zwei Arten der Anforderungsvalidierung konfigurieren:

- In der GET-Methode konfigurieren Sie die Anforderungsvalidierung für URL-Abfragezeichenfolgen-Parameter.

- In der POST-Methode konfigurieren Sie die Anforderungsvalidierung für den Anforderungstext.

Dadurch können nur bestimmte API-Aufrufe an die API weitergeleitet werden.

Laden Sie [die Vorlage zur App-Erstellung für CloudFormation](#) herunter und entpacken Sie sie. Sie verwenden diese Vorlage, um eine unvollständige API zu erstellen. Sie schließen die restlichen Schritte in der API-Gateway-Konsole ab.

Um einen Stack zu erstellen CloudFormation

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **request-validation-tutorial-console** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

So wählen Sie Ihre neu erstellte API aus

1. Wählen Sie den neu erstellten **request-validation-tutorial-console**-Stack aus.
2. Wählen Sie Resources aus.
3. Wählen Sie unter Physische ID Ihre API aus. Dieser Link leitet Sie zur API-Gateway-Konsole weiter.

Bevor Sie die Methoden GET und POST ändern, müssen Sie ein Modell erstellen.

So erstellen Sie ein Modell

1. Ein Modell ist erforderlich, um die Anforderungsvalidierung für den Text einer eingehenden Anfrage zu verwenden. Klicken Sie im Hauptnavigationsbereich auf Modelle, um ein Modell zu erstellen.
2. Wählen Sie Modell erstellen aus.
3. Geben Sie unter Name **PetStoreModel** ein.
4. Geben Sie für Inhaltstyp **application/json** ein. Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
5. Geben Sie für Beschreibung **My PetStore Model** als die Modellbeschreibung ein.
6. Für Modellschema kopieren Sie das folgende Modell in den Code-Editor ein und klicken dann auf Modell erstellen.

```
{
  "type" : "object",
  "required" : [ "name", "price", "type" ],
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "name" : {
      "type" : "string"
    },
    "price" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Weitere Informationen zum Modell finden Sie unter [Datenmodelle für REST APIs](#).

Konfigurieren einer Anforderungsvalidierung für eine **GET**-Methode

1. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die GET-Methode in der Ressourcenstruktur aus.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Wählen Sie für Anforderungs-Validierer die Option Abfragezeichenfolgeparameter und -Header validieren aus.
4. Wählen Sie URL-Abfragezeichenfolgen-Parameter aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
 - b. Geben Sie unter Name **petType** ein.
 - c. Aktivieren Sie die Option Erforderlich.
 - d. Caching bleibt ausgeschaltet.
5. Wählen Sie Speichern.
6. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
7. Wählen Sie URL-Abfragezeichenfolgen-Parameter aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
 - b. Geben Sie unter Name **petType** ein.
 - c. Geben Sie für Zugeordnet von **method.request.querystring.petType** ein. Dadurch wird **petType** dem Typ des Haustiers zugeordnet.

Weitere Informationen zur Datenzuordnung finden Sie [im Tutorial zur Datenzuordnung](#).
 - d. Caching bleibt ausgeschaltet.
8. Wählen Sie Speichern.

So testen Sie die Anforderungsvalidierung für die **GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Abfragezeichenfolgen geben Sie **petType=dog** ein und klicken dann auf Test.
3. Der Methodentest gibt **200 OK** und eine Liste von Hunden zurück.

Informationen zur Umwandlung dieser Ausgabedaten finden Sie im [Tutorial zur Datenzuordnung](#).

4. Entfernen Sie **petType=dog** und wählen Sie Test aus.
5. Der Methodentest gibt einen 400-Fehler mit der folgenden Fehlermeldung zurück:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Konfigurieren der Anforderungsvalidierung für die **POST**-Methode

1. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die POST-Methode aus.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Für Anforderungs-Validator wählen Sie die Option Text validieren aus.
4. Klicken Sie unter Anforderungstext auf Modell hinzufügen.
5. Geben Sie für Content type (Inhaltstyp) **application/json** ein. Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie `$default` ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

Wählen Sie für Modell die Option aus. PetStoreModel

6. Wählen Sie Speichern.

Testen der Anforderungsvalidierung für eine **POST**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Anforderungstext kopieren Sie Folgendes in den Code-Editor:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Wählen Sie Test aus.

3. Der Methodentest gibt 200 OK und eine Erfolgsmeldung zurück.
4. Für Anforderungstext kopieren Sie Folgendes in den Code-Editor:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Wählen Sie Test aus.

5. Der Methodentest gibt einen 400-Fehler mit der folgenden Fehlermeldung zurück:

```
{
  "message": "Invalid request body"
}
```

Der Grund für den ungültigen Anforderungstext wird ganz unten in den Testprotokollen angegeben. In diesem Fall lag der Preis des Haustiers außerhalb des im Modell angegebenen Höchstpreises.

Um einen CloudFormation Stapel zu löschen

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Nächste Schritte

- Informationen zur Transformation von Ausgabedaten und zur Durchführung weiterer Datenzuordnungen finden Sie im [Tutorial zur Datenzuordnung](#).
- Folgen Sie dem Abschnitt [Einrichtung einer einfachen Anforderungsvalidierung mithilfe der AWS CLI](#), um ähnliche Schritte mit der AWS CLI durchzuführen.

Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI

Sie können einen Validator erstellen, um die Anforderungsvalidierung mithilfe der AWS CLI einzurichten. Um diesem Tutorial zu folgen, verwenden Sie eine CloudFormation Vorlage, um eine unvollständige API Gateway zu erstellen.

 Note

Dies ist nicht dieselbe CloudFormation Vorlage wie das Konsolen-Tutorial.

Mithilfe einer vorab bereitgestellten `/validator`-Ressource erstellen Sie die Methoden GET und POST. Beide Methoden werden in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt integriert. Sie konfigurieren die folgenden zwei Anforderungsvalidierungen:

- Für die GET-Methode erstellen Sie einen `params-only`-Validator zur Validierung von URL-Abfragezeichenfolgen-Parametern.
- Für die POST-Methode erstellen Sie einen `body-only`-Validator, um den Anforderungstext zu validieren.

Dadurch können nur bestimmte API-Aufrufe an die API weitergeleitet werden.

Um einen CloudFormation Stapel zu erstellen

Laden Sie [die Vorlage zur App-Erstellung für CloudFormation](#) herunter und entpacken Sie sie.

Um die folgenden Schritte durchzuführen, benötigen Sie die [AWS Command Line Interface \(AWS CLI\) Version 2](#).

Bei langen Befehlen wird ein Escape-Zeichen (`\`) verwendet, um einen Befehl auf mehrere Zeilen aufzuteilen.

 Note

In Windows werden einige Bash-CLI-Befehle, die Sie häufig verwenden (z. B. `zip`), von den integrierten Terminals des Betriebssystems nicht unterstützt. Um eine in Windows integrierte Version von Ubuntu und Bash zu erhalten, [installieren Sie das Windows-Subsystem für Linux](#). Die CLI-Beispielbefehle in diesem Handbuch verwenden die Linux-Formatierung. Befehle, die

Inline-JSON-Dokumente enthalten, müssen neu formatiert werden, wenn Sie die Windows-CLI verwenden.

1. Verwenden Sie den folgenden Befehl, um den CloudFormation Stack zu erstellen.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli  
--template-body file://request-validation-tutorial-cli.zip --capabilities  
CAPABILITY_NAMED_IAM
```

2. CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Verwenden Sie den folgenden Befehl, um den Status Ihres CloudFormation Stacks zu überprüfen.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Wenn der Status Ihres CloudFormation Stacks lautet `StackStatus: "CREATE_COMPLETE"`, verwenden Sie den folgenden Befehl, um relevante Ausgabewerte für future Schritte abzurufen.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli  
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,  
Description: Description}"
```

Die Ausgabewerte sind die folgenden:

- `ApiId`, das ist die ID für die API. Für dieses Tutorial lautet die API-ID `abc123`.
- `ResourceId`, das ist die ID für die Validator-Ressource, in der die POST Methoden GET und verfügbar gemacht werden. Für dieses Tutorial lautet die Ressourcen-ID `efg456`.

So erstellen Sie die Anforderungsvalidierungen und importieren ein Modell

1. Ein Validator ist erforderlich, um die Anforderungsvalidierung mit der AWS CLI zu verwenden. Verwenden Sie den folgenden Befehl, um einen Validator zu erstellen, der nur Anforderungsparameter validiert.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
--no-validate-request-body \  
--validate-request-parameters \  
--name params-only
```


Notieren Sie sich die ID des `params-only`-Validators.

2. Verwenden Sie den folgenden Befehl, um einen Validator zu erstellen, der nur den Anforderungstext validiert.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
  --validate-request-body \  
  --no-validate-request-parameters \  
  --name body-only
```

Notieren Sie sich die ID des `body-only`-Validators.

3. Ein Modell ist erforderlich, um die Anforderungsvalidierung für den Text einer eingehenden Anfrage zu verwenden. Verwenden Sie den folgenden Befehl, um ein Modell zu importieren.

```
aws apigateway create-model --rest-api-id abc123 --name PetStoreModel --description  
'My PetStore Model' --content-type 'application/json' --schema '{"type":  
"object", "required" : [ "name", "price", "type" ], "properties" : { "id" :  
{"type" : "integer"}, "type" : {"type" : "string", "enum" : [ "dog", "cat",  
"fish" ]}, "name" : { "type" : "string"}, "price" : {"type" : "number", "minimum" :  
25.0, "maximum" : 500.0}}}'
```

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

So erstellen Sie die Methoden **GET** und **POST**

1. Rufen Sie den folgenden Befehl auf, um die GET-HTTP-Methode der `/validate`-Ressource hinzuzufügen. Dieser Befehl erstellt die GET-Methode, fügt den `params-only`-Validator hinzu und legt die Abfragezeichenfolge `petType` nach Bedarf fest.

```
aws apigateway put-method --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-validator-id aaa111 \  
  --request-parameters "method.request.querystring.petType=true"
```

Rufen Sie den folgenden Befehl auf, um die POST-HTTP-Methode der /validate-Ressource hinzuzufügen. Mit diesem Befehl wird die POST-Methode erstellt, der body-only-Validator hinzugefügt und das Modell an den Validator angehängt, der nur für den Text gilt.

```
aws apigateway put-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --authorization-type "NONE" \  
    --request-validator-id bbb222 \  
    --request-models 'application/json'=PetStoreModel
```

2. Führen Sie den folgenden Befehl aus, um die 200 OK-Antwort der GET /validate-Methode einzurichten.

```
aws apigateway put-method-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --status-code 200
```

Führen Sie den folgenden Befehl aus, um die 200 OK-Antwort der POST /validate-Methode einzurichten.

```
aws apigateway put-method-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --status-code 200
```

3. Führen Sie den folgenden Befehl aus, um eine Integration mit einem angegebenen HTTP-Endpunkt für die GET /validation-Methode einzurichten.

```
aws apigateway put-integration --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --type HTTP \  
    --integration-http-method GET \  
    --request-parameters '{"integration.request.querystring.type" :  
    "method.request.querystring.petType"}' \  
    --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

Führen Sie den folgenden Befehl aus, um eine Integration mit einem angegebenen HTTP-Endpoint für die POST /validation-Methode einzurichten.

```
aws apigateway put-integration --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --type HTTP \  
    --integration-http-method GET \  
    --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

4. Führen Sie den folgenden Befehl aus, um die Integration der GET /validation-Methode einzurichten.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --status-code 200 \  
    --selection-pattern ""
```

Führen Sie den folgenden Befehl aus, um die Integration der POST /validation-Methode einzurichten.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --status-code 200 \  
    --selection-pattern ""
```

So testen Sie die API-Daten

1. Verwenden Sie den folgenden Befehl, um die GET-Methode zu testen, die eine Anforderungsvalidierung für die Abfragezeichenfolgen durchführt:

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --path-with-query-string '/validate?petType=dog'
```

Das Ergebnis gibt 200 OK und eine Liste von Hunden zurück.

2. Verwenden Sie den folgenden Befehl, um zu testen, ohne die Abfragezeichenfolge `petType` einzuschließen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET
```

Das Ergebnis gibt einen 400-Fehler zurück.

3. Verwenden Sie den folgenden Befehl, um die POST-Methode zu testen, die eine Anforderungsvalidierung für den Anforderungstext durchführt:

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

Das Ergebnis gibt 200 OK und eine Erfolgsmeldung zurück.

4. Verwenden Sie den folgenden Befehl, um mit einem ungültigen Text zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

Das Ergebnis gibt einen 400-Fehler zurück, da der Preis des Hundes über dem vom Modell festgelegten Höchstpreis liegt.

Um einen Stapel zu löschen CloudFormation

- Verwenden Sie den folgenden Befehl, um Ihre CloudFormation Ressourcen zu löschen.

```
aws cloudformation delete-stack --stack-name request-validation-tutorial-cli
```

Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition

Sie können eine Anforderungsvalidierung auf API-Ebene deklarieren, indem Sie eine Reihe von [x-amazon-apigateway-request-Validators.RequestValidator-Objekt](#)-Objekten in der [x-amazon-](#)

[apigateway-request-validators-Objekt](#)-Zuordnung angeben, um auszuwählen, welcher Teil der Anfrage validiert werden soll. In der OpenAPI-Beispieldefinition gibt es zwei Validatoren:

- `all-Validator`, der sowohl den Text anhand des `RequestBodyModel`-Datenmodells als auch die Parameter validiert.

Bei diesem `RequestBodyModel`-Modell muss das JSON-Eingabeobjekt die Eigenschaften `name`, `type` und `price` enthalten. Die Eigenschaft `name` kann eine beliebige Zeichenfolge sein, `type` muss einem der angegebenen Aufzählungsfelder (`["dog", "cat", "fish"]`) entsprechen und `price` muss zwischen 25 und 500 liegen. Der Parameter `id` ist nicht erforderlich.

- `param-only`, der nur die Parameter validiert.

Zum Aktivieren einer Anforderungsvalidierung für alle Methoden einer API geben Sie eine [x-amazon-apigateway-request-validator-Eigenschaft](#) auf API-Ebene in der OpenAPI-Definitionsdatei an. In der Beispiel-OpenAPI-Definition wird der `all` für alle API-Methoden verwendet (sofern nicht anderweitig außer Kraft gesetzt). Wenn ein Modell zur Validierung des Textes verwendet wird und kein übereinstimmender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

Zum Aktivieren einer Anforderungsvalidierung für eine einzelne Methode geben Sie die `x-amazon-apigateway-request-validator`-Eigenschaft auf Methodenebene an. Im Beispiel mit OpenAPI-Definition überschreibt der `param-only`-Validator den `all`-Validator für die GET-Methode.

Um das OpenAPI-Beispiel in API Gateway zu importieren, lesen Sie die folgenden Anweisungen zu [Regionale API in API Gateway importieren](#) oder zu [Edge-optimierte API in API Gateway importieren](#).

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
```

```

        "default" : "/v1"
    }
}
} ],
"paths" : {
    "/validation" : {
        "get" : {
            "parameters" : [ {
                "name" : "q1",
                "in" : "query",
                "required" : true,
                "schema" : {
                    "type" : "string"
                }
            }
        ],
        "responses" : {
            "200" : {
                "description" : "200 response",
                "headers" : {
                    "test-method-response-header" : {
                        "schema" : {
                            "type" : "string"
                        }
                    }
                },
                "content" : {
                    "application/json" : {
                        "schema" : {
                            "$ref" : "#/components/schemas/ArrayOfError"
                        }
                    }
                }
            }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
            "httpMethod" : "GET",
            "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
            "responses" : {
                "default" : {
                    "statusCode" : "400",
                    "responseParameters" : {
                        "method.response.header.test-method-response-header" : "'static
value'"

```

```

        },
        "responseTemplates" : {
            "application/xml" : "xml 400 response template",
            "application/json" : "json 400 response template"
        }
    },
    "2\\d{2}" : {
        "statusCode" : "200"
    }
},
"requestParameters" : {
    "integration.request.querystring.type" : "method.request.querystring.q1"
},
"passthroughBehavior" : "when_no_match",
"type" : "http"
}
},
"post" : {
    "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "schema" : {
            "type" : "string"
        }
    } ],
    "requestBody" : {
        "content" : {
            "application/json" : {
                "schema" : {
                    "$ref" : "#/components/schemas/RequestBodyModel"
                }
            }
        }
    },
    "required" : true
},
"responses" : {
    "200" : {
        "description" : "200 response",
        "headers" : {
            "test-method-response-header" : {
                "schema" : {
                    "type" : "string"
                }
            }
        }
    }
}

```

```

        }
      },
      "content" : {
        "application/json" : {
          "schema" : {
            "$ref" : "#/components/schemas/ArrayOfError"
          }
        }
      }
    },
    "x-amazon-apigateway-request-validator" : "all",
    "x-amazon-apigateway-integration" : {
      "httpMethod" : "POST",
      "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
      "responses" : {
        "default" : {
          "statusCode" : "400",
          "responseParameters" : {
            "method.response.header.test-method-response-header" : "'static
value'"
          },
          "responseTemplates" : {
            "application/xml" : "xml 400 response template",
            "application/json" : "json 400 response template"
          }
        },
        "2\\d{2}" : {
          "statusCode" : "200"
        }
      },
      "requestParameters" : {
        "integration.request.header.custom_h1" : "method.request.header.h1"
      },
      "passthroughBehavior" : "when_no_match",
      "type" : "http"
    }
  }
},
"components" : {
  "schemas" : {
    "RequestBodyModel" : {
      "required" : [ "name", "price", "type" ],

```



```
    "type" : "object",
    "properties" : {
      "id" : {
        "type" : "integer"
      },
      "type" : {
        "type" : "string",
        "enum" : [ "dog", "cat", "fish" ]
      },
      "name" : {
        "type" : "string"
      },
      "price" : {
        "maximum" : 500.0,
        "minimum" : 25.0,
        "type" : "number"
      }
    }
  },
  "ArrayOfError" : {
    "type" : "array",
    "items" : {
      "$ref" : "#/components/schemas/Error"
    }
  },
  "Error" : {
    "type" : "object"
  }
},
"x-amazon-apigateway-request-validators" : {
  "all" : {
    "validateRequestParameters" : true,
    "validateRequestBody" : true
  },
  "params-only" : {
    "validateRequestParameters" : true,
    "validateRequestBody" : false
  }
}
```

OpenAPI 2.0

```

{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0.0",
    "title" : "ReqValidators Sample"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/validation" : {
      "get" : {
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "params-only",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            }
          }
        }
      }
    }
  }
}

```

```

        "responseTemplates" : {
            "application/xml" : "xml 400 response template",
            "application/json" : "json 400 response template"
        }
    },
    "2\\d{2}" : {
        "statusCode" : "200"
    }
},
"requestParameters" : {
    "integration.request.querystring.type" : "method.request.querystring.q1"
},
"passthroughBehavior" : "when_no_match",
"type" : "http"
}
},
"post" : {
    "consumes" : [ "application/json" ],
    "produces" : [ "application/json", "application/xml" ],
    "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "type" : "string"
    }, {
        "in" : "body",
        "name" : "RequestBodyModel",
        "required" : true,
        "schema" : {
            "$ref" : "#/definitions/RequestBodyModel"
        }
    } ],
    "responses" : {
        "200" : {
            "description" : "200 response",
            "schema" : {
                "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
                "test-method-response-header" : {
                    "type" : "string"
                }
            }
        }
    }
}

```

```

    },
    "x-amazon-apigateway-request-validator" : "all",
    "x-amazon-apigateway-integration" : {
      "httpMethod" : "POST",
      "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
      "responses" : {
        "default" : {
          "statusCode" : "400",
          "responseParameters" : {
            "method.response.header.test-method-response-header" : "'static
value'"
          },
          "responseTemplates" : {
            "application/xml" : "xml 400 response template",
            "application/json" : "json 400 response template"
          }
        },
        "2\\d{2}" : {
          "statusCode" : "200"
        }
      },
      "requestParameters" : {
        "integration.request.header.custom_h1" : "method.request.header.h1"
      },
      "passthroughBehavior" : "when_no_match",
      "type" : "http"
    }
  }
},
"definitions" : {
  "RequestBodyModel" : {
    "type" : "object",
    "required" : [ "name", "price", "type" ],
    "properties" : {
      "id" : {
        "type" : "integer"
      },
      "type" : {
        "type" : "string",
        "enum" : [ "dog", "cat", "fish" ]
      },
      "name" : {
        "type" : "string"
      }
    }
  }
}

```

```
    },
    "price" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  },
  "ArrayOfError" : {
    "type" : "array",
    "items" : {
      "$ref" : "#/definitions/Error"
    }
  },
  "Error" : {
    "type" : "object"
  }
},
"x-amazon-apigateway-request-validators" : {
  "all" : {
    "validateRequestParameters" : true,
    "validateRequestBody" : true
  },
  "params-only" : {
    "validateRequestParameters" : true,
    "validateRequestBody" : false
  }
}
}
```

AWS CloudFormation Vorlage einer Beispiel-API mit grundlegender Anforderungsvalidierung

Die folgende CloudFormation Beispielvorlagendefinition definiert eine Beispiel-API mit aktivierter Anforderungsvalidierung. Die [PetStore-API ist eine Teilmenge der](#) -API. Mithilfe einer POST-Methode wird ein Haustier zur Sammlung pets hinzugefügt und mit einer GET-Methode erfolgt die Abfrage der Haustiere nach Typ.

Es wurden zwei Anforderungsvalidatoren deklariert:

GETValidator

Dieser Validator ist für die GET-Methode aktiviert. Damit kann API Gateway überprüfen, ob der erforderliche Abfrageparameter (q1) in der eingehenden Anfrage enthalten und nicht leer ist.

POSTValidator

Dieser Validator ist für die POST-Methode aktiviert. Damit kann API Gateway prüfen, ob das Nutzlast-Anforderungsformat dem angegebenen `RequestBodyModel` entspricht. Wenn der Inhaltstyp `application/json` ist und kein passender Inhaltstyp gefunden wird, wird die Anforderungsüberprüfung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie `$default` an. `RequestBodyModel` enthält ein zusätzliches Modell, `RequestBodyModelId`, um die Pet-ID zu definieren.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: ReqValidatorsSample
  RequestBodyModelId:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModelId
        properties:
          id:
            type: integer
  RequestBodyModel:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
```

Description: Request body model for Pet type, name, price, and ID.

Schema:

\$schema: 'http://json-schema.org/draft-04/schema#'

title: RequestBodyModel

required:

- price
- name
- type

type: object

properties:

id:

"\$ref": !Sub

- 'https://apigateway.amazonaws.com/restapis/\${Api}/models/
\${RequestBodyModelId}'

- Api: !Ref Api

RequestBodyModelId: !Ref RequestBodyModelId

price:

type: number

minimum: 25

maximum: 500

name:

type: string

type:

type: string

enum:

- "dog"
- "cat"
- "fish"

GETValidator:

Type: AWS::ApiGateway::RequestValidator

Properties:

Name: params-only

RestApiId: !Ref Api

ValidateRequestBody: False

ValidateRequestParameters: True

POSTValidator:

Type: AWS::ApiGateway::RequestValidator

Properties:

Name: body-only

RestApiId: !Ref Api

ValidateRequestBody: True

ValidateRequestParameters: False

ValidationResource:

Type: 'AWS::ApiGateway::Resource'

```

Properties:
  RestApiId: !Ref Api
  ParentId: !GetAtt Api.RootResourceId
  PathPart: 'validation'
ValidationMethodGet:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref ValidationResource
    HttpMethod: GET
    AuthorizationType: NONE
    RequestValidatorId: !Ref GETValidator
    RequestParameters:
      method.request.querystring.q1: true
    Integration:
      Type: HTTP_PROXY
      IntegrationHttpMethod: GET
      Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
ValidationMethodPost:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref ValidationResource
    HttpMethod: POST
    AuthorizationType: NONE
    RequestValidatorId: !Ref POSTValidator
    RequestModels:
      application/json : !Ref RequestBodyModel
    Integration:
      Type: HTTP_PROXY
      IntegrationHttpMethod: POST
      Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
ApiDeployment:
  Type: 'AWS::ApiGateway::Deployment'
  DependsOn:
    - ValidationMethodGet
    - RequestBodyModel
  Properties:
    RestApiId: !Ref Api
    StageName: !Sub '${StageName}'
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'

```

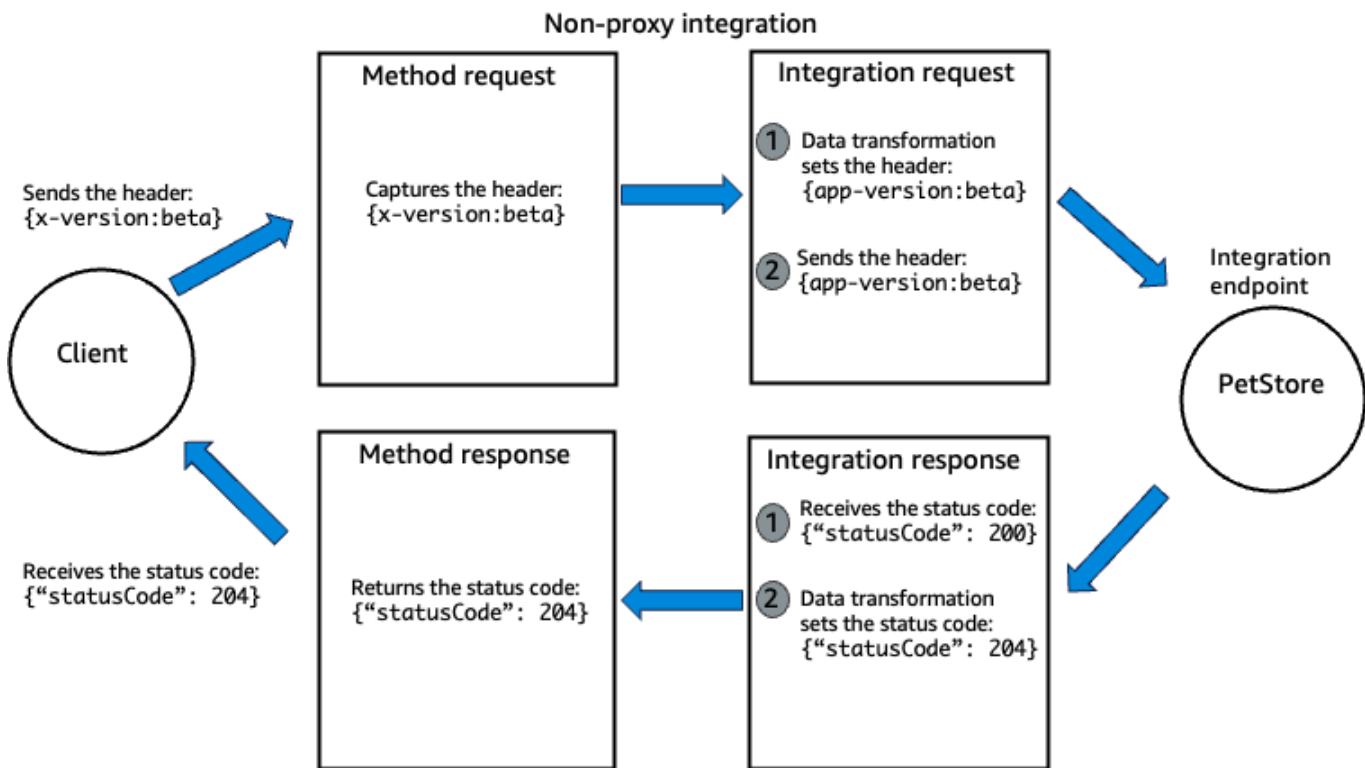

Datentransformationen für REST-APIs in API Gateway

Note

In diesem Abschnitt werden die Features erläutert, die Sie mit einer Nicht-Proxy-Integration verwenden. Wir empfehlen jedoch, wann immer möglich, eine Proxy-Integration für Ihre REST-API zu verwenden. Eine Proxy-Integration bietet eine vereinfachte Integrationskonfiguration und kann sich gemeinsam mit dem Backend weiterentwickeln, ohne die bestehende Einrichtung zu beeinträchtigen. Weitere Informationen finden Sie unter [the section called “Auswahl eines API-Integrationstyps”](#).

Wenn Sie eine Nicht-Proxy-Integration verwenden, können Sie zwei Features von API Gateway nutzen, um Ihre Methodenanforderung und Ihre Integrationsantwort zu transformieren. Sie können die Methodenanforderung transformieren, wenn sie ein anderes Format der Nutzdaten als die Integrationsanforderung verwendet. Sie können die Integrationsantwort transformieren, wenn sie ein anderes Format der Nutzdaten zurückliefert als das Format, das Sie in der Methodenantwort zurückgeben müssen. Weitere Informationen zum Lebenszyklus einer Anfrage finden Sie unter [the section called “Beispielressource für eine REST-API”](#).

Im folgenden Beispiel wird eine Datentransformation dargestellt: Für den Header "x-version:beta" wird der Header-Parameter x-version in den Header-Parameter app-version transformiert. Diese Transformation von x-version in app-version erfolgt in der Integrationsanforderung. Auf diese Weise empfängt das Integrationsendpunkt den transformierten Header-Parameterwert. Wenn der Integrationsendpunkt einen Statuscode zurückgibt, wird dieser von 200 in 204 transformiert, bevor er als Methodenantwort zurückgegeben wird.



Um eine Datentransformation zu erstellen, können Sie die folgenden Features verwenden:

Parameterzuweisung

Bei der Zuweisung von Parametern können Sie die URL-Pfadparameter, die URL-Abfragezeichenfolgenparameter oder die HTTP-Header-Werte der Integrationsanforderung ändern, aber nicht die Nutzdaten der Integrationsanforderung. Sie können auch die Header-Werte der HTTP-Antwort ändern. Verwenden Sie die Parameterzuweisung, um statische Header-Werte für Cross-Origin Resource Sharing (CORS) zu erstellen.

Sie können die Zuweisung von Parametern in Ihrer Integrationsanforderung sowohl für Proxy- als auch für Nicht-Proxy-Integrationen verwenden. Um jedoch die Zuweisung von Parametern für eine Integrationsantwort zu nutzen, benötigen Sie eine Nicht-Proxy-Integration. Die Zuweisung von Parametern erfordert kein Scripting in der [Velocity Template Language \(VTL\)](#). Weitere Informationen finden Sie unter [the section called "Parameterzuweisung"](#).

Transformationen mit Zuweisungsvorlagen

Bei Transformationen mit Zuweisungsvorlagen verwenden Sie eine Vorlage, um URL-Pfadparameter, URL-Abfragezeichenfolgenparameter, HTTP-Header sowie den Body der Integrationsanforderung oder Integrationsantwort zuzuweisen. Eine Zuweisungsvorlage ist ein

Skript in der [Velocity Template Language \(VTL\)](#), das [JSONPath-Ausdrücke](#) nutzt und auf die Nutzdaten angewendet wird, basierend auf dem Content-type-Header.

Mit einer Zuweisungsvorlage können Sie Folgendes tun:

- Wählen Sie aus, welche Daten mithilfe der Integration gesendet werden sollen AWS-Services, z. B. mit Amazon DynamoDB- oder Lambda-Funktionen oder HTTP-Endpunkten. Weitere Informationen finden Sie unter [the section called “Tutorial: Ändern Sie die Integrationsanfrage und die Antwort für Integrationen zu AWS service”](#).
- Bedingt die Parameter einer Integrationsanforderung und -antwort einer API überschreiben, neue Header-Werte erstellen und Statuscodes überschreiben. Weitere Informationen finden Sie unter [the section called “Überschreiben der Anforderungs- und Antwortparameter und Statuscodes Ihrer API”](#).

Das Verhalten Ihrer API festlegen, wenn die Nutzdaten einer Integrationsanforderung einen Content-type-Header enthalten, für den keine passenden Zuweisungsvorlagen vorhanden sind. Dies wird als Integrations-Passthrough-Verhalten bezeichnet. Weitere Informationen finden Sie unter [the section called “Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen”](#).

Auswahl zwischen Zuweisung von Parametern und Transformationen mit Zuweisungsvorlagen

Wir empfehlen, die Zuweisung von Parametern zu verwenden, um Daten nach Möglichkeit zu transformieren. Wenn Ihre API jedoch erfordert, dass Sie den Body ändern oder bedingte Überschreibungen und Modifikationen basierend auf der eingehenden Integrationsanforderung oder Integrationsantwort vornehmen und Sie keine Proxy-Integration verwenden können, sollten Sie Transformationen mit Zuweisungsvorlagen einsetzen.

Parameterzuweisung für REST-APIs in API Gateway

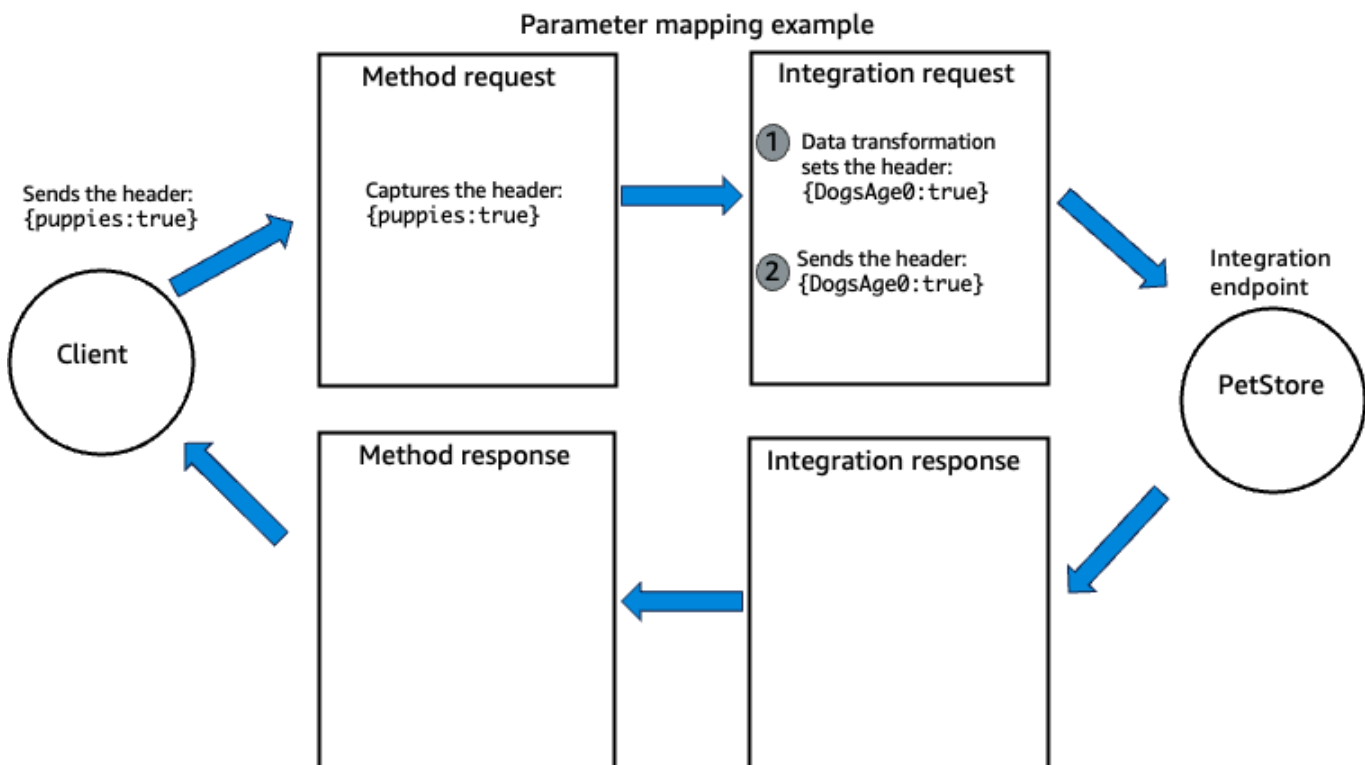
Note

Wenn Sie eine HTTP-API verwenden, finden Sie weitere Informationen unter [the section called “Parameterzuweisung”](#).

Bei der Zuweisung von Parametern weisen Sie Anfrage- oder Antwortparameter zu. Sie können Parameter mithilfe von Ausdrücken zur Parameterzuweisung oder statischen Werten zuweisen. Eine

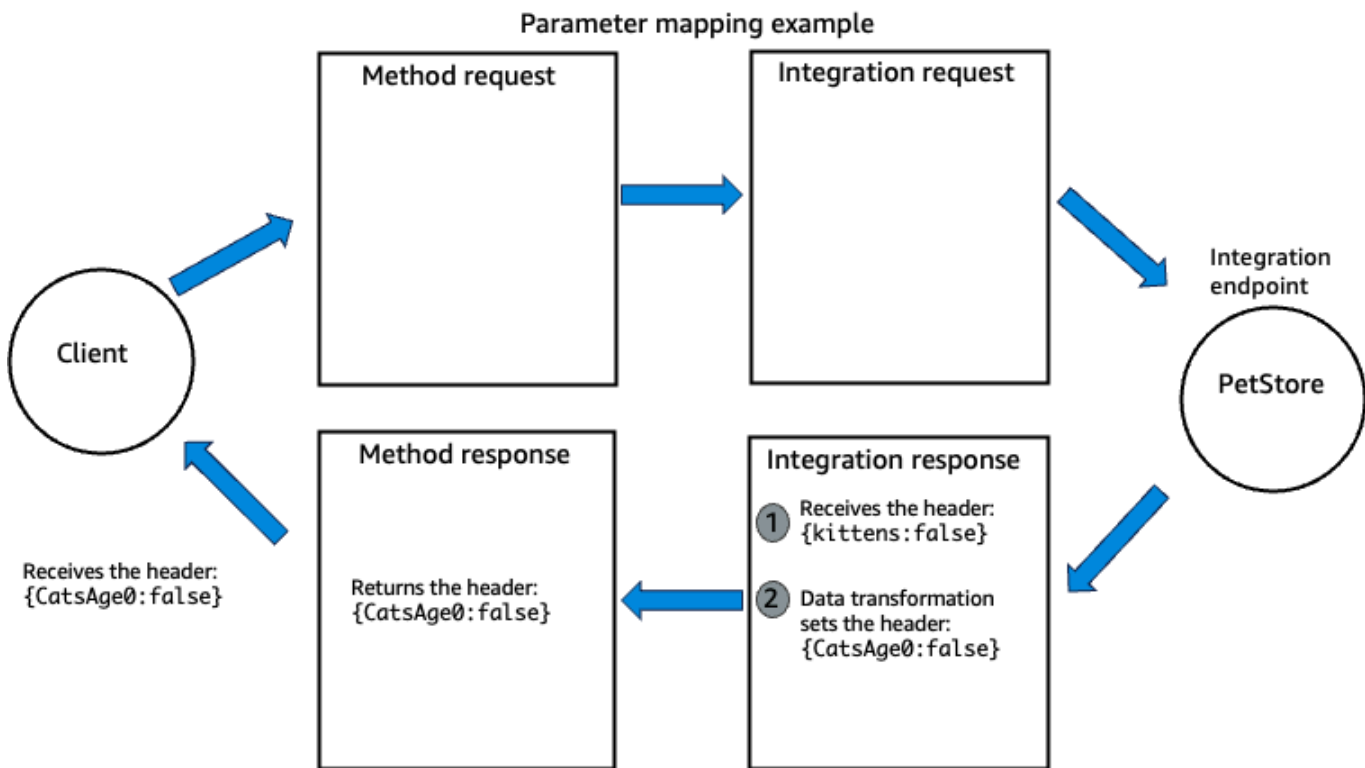
Liste der Zuweisungsweissungsausdrücke finden Sie unter [the section called “Quellenreferenz für Parameterzuweisungen”](#). Sie können die Zuweisung von Parametern in Ihrer Integrationsanforderung sowohl für Proxy- als auch für Nicht-Proxy-Integrationen verwenden. Um jedoch die Zuweisung von Parametern für eine Integrationsantwort zu nutzen, benötigen Sie eine Nicht-Proxy-Integration.

Beispielsweise können Sie den Header-Parameter `puppies` der Methodenanforderung dem Header-Parameter `DogsAge0` der Integrationsanforderung zuweisen. Wenn ein Client den Header `puppies:true` an Ihre API sendet, sendet die Integrationsanforderung den Anfrage-Header `DogsAge0:true` an den Integrationsendpunkt. Das folgende Diagramm zeigt den Anfragelebenszyklus dieses Beispiels.



Informationen zum Erstellen dieses Beispiels mit API Gateway finden Sie unter [the section called “Beispiel 1: Zuweisen eines Methodenanforderungsparameters zu einem Integrationsanforderungsparameter”](#).

Als weiteres Beispiel können Sie den Header-Parameter `kittens` der Integrationsantwort dem Header-Parameter `CatsAge0` der Methodenantwort zuweisen. Wenn der Integrationsendpunkt dann `kittens:false` zurückgibt, erhält der Client den Header `CatsAge0:false`. Das folgende Diagramm zeigt den Anfragelebenszyklus dieses Beispiels.



Themen

- [Beispiele für Parameterzuweisungen für REST-APIs in API Gateway](#)
- [Quellenreferenz für Parameterzuweisungen für REST-APIs in API Gateway](#)

Beispiele für Parameterzuweisungen für REST-APIs in API Gateway

Die folgenden Beispiele zeigen, wie Sie Ausdrücke zur Parameterzuweisung mithilfe der API-Gateway-Konsole, OpenAPI oder CloudFormation -Vorlagen erstellen. Ein Beispiel, wie Sie die Parameterzuweisung verwenden, um die erforderlichen CORS-Header zu erstellen, finden Sie unter [the section called "CORES"](#).

Beispiel 1: Zuweisen eines Methodenanforderungsparameters zu einem Integrationsanforderungsparameter

Im folgenden Beispiel wird der Header-Parameter `puppies` der Methodenanforderung dem Header-Parameter `DogsAge0` der Integrationsanforderung zugewiesen.

AWS-Managementkonsole

So weisen Sie den Parameter der Methodenanforderung zu

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie eine Methode aus.

Ihre Methode muss eine Nicht-Proxy-Integration aufweisen.

4. Wählen Sie unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
5. Wählen Sie HTTP-Anfrage-Header aus.
6. Wählen Sie Add header.
7. Geben Sie unter Name **puppies** ein.
8. Wählen Sie Speichern.
9. Wählen Sie die Registerkarte Integrationsanfrage aus und klicken Sie dann unter Einstellungen für Integrationsanforderungen auf Bearbeiten.

Das fügt AWS-Managementkonsole automatisch eine Parameterzuordnung von `method.request.header.puppies` bis `puppies` für Sie hinzu, aber Sie müssen den Namen ändern, damit er dem Anforderungsheader-Parameter entspricht, der von Ihrem Integrationsendpunkt erwartet wird.

10. Geben Sie unter Name **DogsAge0** ein.
11. Wählen Sie Speichern.
12. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Nachfolgend wird beschrieben, wie Sie überprüfen können, ob Ihre Parameterzuordnung erfolgreich war.

Überprüfen der erfolgreichen Parameterzuweisung (optional)

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Geben Sie unter Headers **puppies:true** ein.
3. Wählen Sie Test aus.

4. Unter Protokoll sollte das Ergebnis wie folgt aussehen:

```
Tue Feb 04 00:28:36 UTC 2025 : Method request headers: {puppies=true}
Tue Feb 04 00:28:36 UTC 2025 : Method request body before transformations:
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request headers: {DogsAge0=true,
  x-amzn-apigateway-api-id=abcd1234, Accept=application/json, User-
Agent=AmazonAPIGateway_aaaaaaa, X-Amzn-Trace-Id=Root=1-abcd-12344}
```

Der Header-Parameter der Anfrage wurde von puppies auf DogsAge0 geändert.

CloudFormation

In diesem Beispiel verwenden Sie die [Body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: ParameterMappingExample
          version: "2025-02-04T00:30:41Z"
        paths:
          /pets:
            get:
              parameters:
                - name: puppies
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
```

```

        default:
            statusCode: "200"
        requestParameters:
            integration.request.header.DogsAge0: method.request.header.puppies
        passthroughBehavior: when_no_match
        type: http
    ApiGatewayDeployment:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    ApiGatewayDeployment20250219:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    Stage:
        Type: 'AWS::ApiGateway::Stage'
        Properties:
            DeploymentId: !Ref ApiGatewayDeployment20250219
            RestApiId: !Ref Api
            StageName: prod

```

OpenAPI

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ParameterMappingExample",
    "version" : "2025-02-04T00:30:41Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "puppies",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }
      ],
      "responses" : {
        "200" : {

```



```

        "description" : "200 response"
      },
    },
    "x-amazon-apigateway-integration" : {
      "httpMethod" : "GET",
      "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
      "responses" : {
        "default" : {
          "statusCode" : "200"
        }
      },
    },
    "requestParameters" : {
      "integration.request.header.DogsAge0" : "method.request.header.puppies"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "http"
  }
}
}
}
}
}

```

Beispiel 2: Zuweisen mehrerer Parameter der Methodenanforderung zu verschiedenen Parametern der Integrationsanforderung

Im folgenden Beispiel wird der Mehrwert-Abfragezeichenfolgenparameter der Methodenanforderung `methodRequestQueryParam` dem Abfragezeichenfolgenparameter der Integrationsanforderung `integrationQueryParam` zugewiesen, und der Header-Parameter der Methodenanforderung `methodRequestHeaderParam` wird dem Pfadparameter der Integrationsanforderung `integrationPathParam` zugewiesen.

AWS-Managementkonsole

So weisen Sie Methodenanforderungsparameter zu

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie eine Methode aus.

Ihre Methode muss eine Nicht-Proxy-Integration aufweisen.

4. Wählen Sie unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
5. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.
6. Wählen Sie Abfragezeichenfolge hinzufügen aus.
7. Geben Sie unter Name **methodRequestQueryParam** ein.
8. Wählen Sie HTTP-Anfrage-Headers aus.
9. Wählen Sie Add header.
10. Geben Sie unter Name **methodRequestHeaderParam** ein.
11. Wählen Sie Speichern.
12. Wählen Sie die Registerkarte Integrationsanfrage aus und klicken Sie dann unter Einstellungen für Integrationsanforderungen auf Bearbeiten.
13. Klicken Sie auf URL-Pfadparameter.
14. Klicken Sie auf Pfadparameter hinzufügen.
15. Geben Sie unter Name **integrationPathParam** ein.
16. Geben Sie für Zugeordnet von **method.request.header.methodRequestHeaderParam** ein.

Damit wird der von Ihnen in der Methodenanforderung angegebene Header-Parameter einem neuen Pfadparameter der Integrationsanforderung zugewiesen.

17. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.
18. Wählen Sie Abfragezeichenfolge hinzufügen aus.
19. Geben Sie unter Name **integrationQueryParam** ein.
20. Geben Sie für Zugeordnet von **method.request.multivaluequerystring.methodRequestQueryParam** ein.

Damit wird der Parameter der Mehrwert-Anfragezeichenfolge einem neuen einfachwertigen Parameter der Mehrwert-Anfragezeichenfolge der Integrationsanforderung zugewiesen.

21. Wählen Sie Speichern.
22. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

CloudFormation

In diesem Beispiel verwenden Sie die [body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

Die folgende OpenAPI-Definition erstellt die folgenden Parameterzuordnungen für eine HTTP-Integration:

- den Header der Methodenanforderung, mit dem Namen `methodRequestHeaderParam`, in den Pfadparameter der Integrationsanforderung, mit dem Namen `integrationPathParam`
- die Mehrwert-Anfragezeichenfolge der Methodenanforderung, mit dem Namen `methodRequestQueryParam`, in die Anfragezeichenfolge der Integrationsanforderung, mit dem Namen `integrationQueryParam`

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 2
          version: "2025-01-15T19:12:31Z"
        paths:
          /:
            post:
              parameters:
                - name: methodRequestQueryParam
                  in: query
                  schema:
                    type: string
                - name: methodRequestHeaderParam
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:

```

```

        integration.request.querystring.integrationQueryParam:
method.request.multivaluequerystring.methodRequestQueryParam
        integration.request.path.integrationPathParam:
method.request.header.methodRequestHeaderParam
    requestTemplates:
        application/json: '{"statusCode": 200}'
    passthroughBehavior: when_no_templates
    timeoutInMillis: 29000
    type: http
ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
        RestApiId: !Ref Api
ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
        RestApiId: !Ref Api
Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
        DeploymentId: !Ref ApiGatewayDeployment20250219
        RestApiId: !Ref Api
        StageName: prod

```

OpenAPI

Die folgende OpenAPI-Definition erstellt die folgenden Parameterzuordnungen für eine HTTP-Integration:

- den Header der Methodenanforderung, mit dem Namen `methodRequestHeaderParam`, in den Pfadparameter der Integrationsanforderung, mit dem Namen `integrationPathParam`
- die Mehrwert-Anfragezeichenfolge der Methodenanforderung, mit dem Namen `methodRequestQueryParam`, in die Anfragezeichenfolge der Integrationsanforderung, mit dem Namen `integrationQueryParam`

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example 2",

```

```

    "version" : "2025-01-15T19:12:31Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "parameters" : [ {
          "name" : "methodRequestQueryParam",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "methodRequestHeaderParam",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.integrationQueryParam" :
"method.request.multivaluequerystring.methodRequestQueryParam",
            "integration.request.path.integrationPathParam" :
"method.request.header.methodRequestHeaderParam"
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000,
          "type" : "http"
        }
      }
    }
  }
}

```

```
}  
}  
}  
}
```

Beispiel 3: Zuweisen von Feldern aus dem JSON-Body der Anfrage zu Parametern der Integrationsanforderung

Sie können Integrationsanforderungsparameter auch aus Feldern im JSON-Body der Anfrage mithilfe eines [JSONPath-Ausdrucks](#) zuweisen. Im folgenden Beispiel wird der Body der Methodenanforderung einem Header in einer Integrationsanforderung mit dem Namen `body-header` zugewiesen und ein Teil des Anfragebodys, ausgedrückt durch einen JSON-Ausdruck, wird einem Header in einer Integrationsanforderung mit dem Namen `pet-price` zugewiesen.

Um dieses Beispiel zu testen, geben Sie bitte eine Eingabe ein, die eine Preiskategorie enthält, wie beispielsweise die folgende:

```
[  
  {  
    "id": 1,  
    "type": "dog",  
    "price": 249.99  
  }  
]
```

AWS-Managementkonsole

So weisen Sie Methodenanforderungsparameter zu

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie eine POST-, PUT-, PATCH- oder ANY-Methode aus.

Ihre Methode muss eine Nicht-Proxy-Integration aufweisen.

4. Wählen Sie unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
5. Klicken Sie auf URL-Anforderungsheader-Parameter.
6. Klicken Sie auf Header-Parameter der Anfrage hinzufügen.

7. Geben Sie unter Name **body-header** ein.
8. Geben Sie für Zugeordnet von **method.request.body** ein.

Damit wird der Body der Methodenanforderung einem neuen Header-Parameter der Integrationsanforderung zugewiesen.

9. Klicken Sie auf Header-Parameter der Anfrage hinzufügen.
10. Geben Sie unter Name **pet-price** ein.
11. Geben Sie für Zugeordnet von **method.request.body[0].price** ein.

Damit wird ein Teil des Bodys der Methodenanforderung einem neuen Header-Parameter der Integrationsanforderung zugewiesen.

12. Wählen Sie Speichern.
13. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

CloudFormation

In diesem Beispiel verwenden Sie die [body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 3
          version: "2025-01-15T19:19:14Z"
        paths:
          /:
            post:
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
```

```

        statusCode: "200"
        requestParameters:
            integration.request.header.pet-price: method.request.body[0].price
            integration.request.header.body-header: method.request.body
        requestTemplates:
            application/json: '{"statusCode": 200}'
        passthroughBehavior: when_no_templates
        timeoutInMillis: 29000
        type: http
    ApiGatewayDeployment:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    ApiGatewayDeployment20250219:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    Stage:
        Type: 'AWS::ApiGateway::Stage'
        Properties:
            DeploymentId: !Ref ApiGatewayDeployment20250219
            RestApiId: !Ref Api
            StageName: prod

```

OpenAPI

Die folgende OpenAPI-Definition weist Integrationsanforderungsparameter aus Feldern im JSON-Body der Anfrage zu.

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example 3",
    "version" : "2025-01-15T19:19:14Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
          "200" : {
            "description" : "200 response"

```



```

    }
  },
  "x-amazon-apigateway-integration" : {
    "httpMethod" : "GET",
    "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "responses" : {
      "default" : {
        "statusCode" : "200"
      }
    },
    "requestParameters" : {
      "integration.request.header.pet-price" : "method.request.body[0].price",
      "integration.request.header.body-header" : "method.request.body"
    },
    "requestTemplates" : {
      "application/json" : "{\"statusCode\": 200}"
    },
    "passthroughBehavior" : "when_no_templates",
    "timeoutInMillis" : 29000,
    "type" : "http"
  }
}
}
}
}
}

```

Beispiel 4: Zuweisen der Integrationsantwort zur Methodenantwort

Sie können auch die Integrationsantwort der Methodenantwort zuweisen. Im folgenden Beispiel wird der Body der Integrationsantwort einem Header der Methodenantwort mit dem Namen `location` zugewiesen, der Header der Integrationsantwort `x-app-id` dem Header der Methodenantwort `id` zugewiesen und der Header der Mehrwert-Integrationsantwort `item` dem Header der Methodenantwort `items` zugewiesen.

AWS-Managementkonsole

So weisen Sie die Integrationsantwort zu

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine REST-API aus.

3. Wählen Sie eine Methode aus.

Ihre Methode muss eine Nicht-Proxy-Integration aufweisen.

4. Wechseln Sie zur Registerkarte Methodenantwort und wählen Sie unter Antwort 200 die Option Bearbeiten aus.
5. Wählen Sie unter Header-Name die Option Header hinzufügen aus.
6. Erstellen Sie drei Header mit den Namen **id**, **item** und **location**.
7. Wählen Sie Speichern aus.
8. Wechseln Sie zur Registerkarte Integrationsantwort und wählen Sie unter Standard – Antwort die Option Bearbeiten aus.
9. Geben Sie unter Header-Zuweisungen Folgendes ein.
 - a. Geben Sie unter id **integration.response.header.x-app-id** ein.
 - b. Geben Sie unter item **integration.response.multivalueheader.item** ein.
 - c. Geben Sie unter location **integration.response.body.redirect.url** ein.
10. Wählen Sie Speichern aus.
11. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

CloudFormation

In diesem Beispiel verwenden Sie die [body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example
          version: "2025-01-15T19:21:35Z"
        paths:
          /:
            post:
              responses:
                "200":
```

```

        description: 200 response
        headers:
          item:
            schema:
              type: string
          location:
            schema:
              type: string
          id:
            schema:
              type: string
        x-amazon-apigateway-integration:
          type: http
          httpMethod: GET
          uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
          responses:
            default:
              statusCode: "200"
              responseParameters:
                method.response.header.id: integration.response.header.x-app-
id
                method.response.header.location:
integration.response.body.redirect.url
                method.response.header.item:
integration.response.multivalueheader.item
              requestTemplates:
                application/json: '{"statusCode": 200}'
              passthroughBehavior: when_no_templates
              timeoutInMillis: 29000
      ApiGatewayDeployment:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
          RestApiId: !Ref Api
      ApiGatewayDeployment20250219:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
          RestApiId: !Ref Api
      Stage:
        Type: 'AWS::ApiGateway::Stage'
        Properties:
          DeploymentId: !Ref ApiGatewayDeployment20250219
          RestApiId: !Ref Api

```

```
StageName: prod
```

OpenAPI

Die folgende OpenAPI-Definition weist die Integrationsantwort der Methodenantwort zu.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example",
    "version" : "2025-01-15T19:21:35Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "item" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "location" : {
              "schema" : {
                "type" : "string"
              }
            },
            "id" : {
              "schema" : {
                "type" : "string"
              }
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-integration" : {
    "type" : "http",
    "httpMethod" : "GET",
    "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "responses" : {
      "default" : {
        "statusCode" : "200",
```

```
        "responseParameters" : {
            "method.response.header.id" : "integration.response.header.x-app-
id",
            "method.response.header.location" :
"integration.response.body.redirect.url",
            "method.response.header.item" :
"integration.response.multivalueheader.item"
        }
    },
    "requestTemplates" : {
        "application/json" : "{\"statusCode\": 200}"
    },
    "passthroughBehavior" : "when_no_templates",
    "timeoutInMillis" : 29000
}
}
```

Quellenreferenz für Parameterzuweisungen für REST-APIs in API Gateway

Wenn Sie eine Parameterzuweisung erstellen, geben Sie die zu ändernden Parameter der Methodenanforderung oder Integrationsantwort an und legen fest, wie diese Parameter geändert werden sollen.

Die folgende Tabelle zeigt die Parameter der Methodenanforderung, die Sie zuweisen können, sowie den Ausdruck, um die Zuweisung zu erstellen. In diesen Ausdrücken *name* steht der Name eines Methodenanforderungsparameters. Beispielsweise verwenden Sie zur Zuweisung des Header-Parameters in der Anforderung `puppies` den Ausdruck `method.request.header.puppies`. Ihr Ausdruck muss dem regulären Ausdruck `'^[a-zA-Z0-9._$-]+$'` entsprechen. Sie können die Parameterzuweisung in Ihrer Integrationsanforderung sowohl für Proxy- als auch für Nicht-Proxy-Integrationen verwenden.

Zugewiesene Datenquelle	Mapping-Ausdruck
Methodenanforderungspfad	<code>method.request.path.</code> <i>name</i>
Abfragezeichenfolge der Methodenanforderung	<code>method.request.querystring.</code> <i>name</i>

Zugewiesene Datenquelle	Mapping-Ausdruck
Multi-value Abfragezeichenfolge für eine Methodenanfrage	<code>method.request.multivaluequerystring. <i>name</i></code>
Methodenanforderungs-Header	<code>method.request.header. <i>name</i></code>
Multi-value Header der Methodenanforderung	<code>method.request.multivalueheader. <i>name</i></code>
Methodenanforderungstext	<code>method.request.body</code>
Hauptteil der Methodenanforderung (JsonPath)	<code>method.request.body. <i>JSONPath_EXPRESSION</i></code> . <i>JSONPath_EXPRESSION</i> ist ein JSONPath-Ausdruck für ein JSON-Feld im Hauptteil einer Anfrage. Weitere Informationen finden Sie unter JSONPath expression .
Stufenvariablen	<code>stageVariables. <i>name</i></code>
Kontextvariablen	<code>context.<i>name</i></code> Der Name muss eine der unterstützten Kontextvariablen sein.
Statischer Wert	<code>'<i>static_value</i>'</code> . Das <i>static_value</i> ist ein Zeichenkettenliteral und muss in zwei einfache Anführungszeichen eingeschlossen werden. Beispiel, <code>'https://www.example.com'</code> .

Die folgende Tabelle zeigt die Parameter der Integrationsantwort, die Sie zuweisen können, sowie den Ausdruck zur Erstellung der Zuweisung. *name* Ist in diesen Ausdrücken der Name eines Antwortparameters für die Integration. Sie können Header der Methodenantwort aus beliebigen Headern der Integrationsantwort oder dem Body einer Integrationsantwort, aus \$context-

Variablen oder aus statischen Werten zuweisen. Um die Zuweisung von Parametern für eine Integrationsantwort zu verwenden, benötigen Sie eine Nicht-Proxy-Integration.

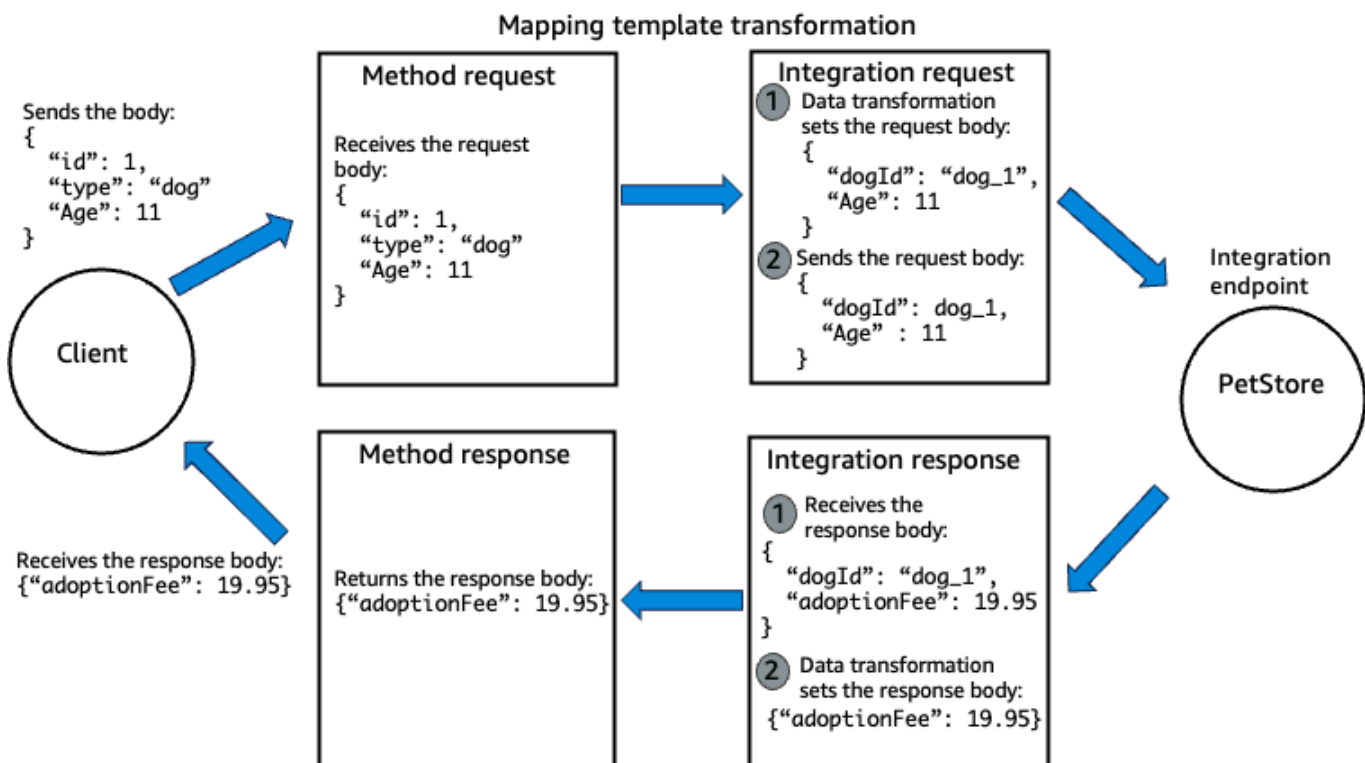
Zugewiesene Datenquelle	Mapping-Ausdruck
Integrationsantwort-Header	<code>integration.response.header. <i>name</i></code>
Integrationsantwort-Header	<code>integration.response.multivalueheader. <i>name</i></code>
Integrationsantworttext	<code>integration.response.body</code>
Hauptteil der Integrationsantwort (JsonPath)	<code>integration.response.body. <i>JSONPath_EXPRESSION</i></code> <i>JSONPath_EXPRESSION</i> ist ein JSONPath-Ausdruck für ein JSON-Feld im Hauptteil einer Antwort. Weitere Informationen finden Sie unter JSONPath expression .
Stufenvariable	<code>stageVariables. <i>name</i></code>
Kontextvariable	<code>context.<i>name</i></code> Der name muss eine der unterstützten Kontextvariablen sein.
Statischer Wert	<code>'<i>static_value</i>'</code> Das <i>static_value</i> ist ein Zeichenkettenliteral und muss in zwei einfache Anführungszeichen eingeschlossen werden. Beispiel, <code>'https://www.example.com'</code> .

Transformationen mit Zuweisungsvorlagen für REST-APIs in API Gateway

Eine Transformation mit Zuweisungsvorlagen verwendet eine Zuweisungsvorlage, um Ihre Integrationsanforderung oder Integrationsantwort zu ändern. Eine Zuweisungsvorlage ist ein in [Velocity Template Language \(VTL\)](#) ausgedrücktes Skript, das mithilfe von [JSONPath](#)-Ausdrücken

auf die Nutzdaten angewendet wird, basierend auf dem Content-type-Header. Sie verwenden Zuweisungsvorlagen, wenn Sie Transformationen mit Zuweisungsvorlagen einsetzen. In diesem Abschnitt werden konzeptionelle Informationen zu Zuweisungsvorlagen beschrieben.

Das folgende Diagramm zeigt den Anforderungslebenszyklus für eine POST /pets Ressource, die über eine Integration mit einem PetStore Integrationsendpunkt verfügt. In dieser API senden Benutzer Daten zu einem Haustier, und der Integrationsendpunkt gibt die mit dem Haustier verbundene Adoptionsgebühr zurück. In diesem Anfragelebenszyklus filtern Transformationen mit Zuweisungsvorlagen den Anfragebody zum Integrationsendpunkt und den Antwortbody vom Integrationsendpunkt.



Die folgenden Abschnitte erläutern den Lebenszyklus von Anfrage und Antwort.

Methodenanforderung und Integrationsanforderung

Im vorherigen Beispiel gilt: Wenn dies der Anforderungs-Body ist, der an die Methodenanforderung gesendet wird:

```
POST /pets
HTTP/1.1
Host:abcd1234.us-west-2.amazonaws.com
```



```
Content-type: application/json
```

```
{
  "id": 1,
  "type": "dog",
  "Age": 11,
}
```

Dieser Body der Anfrage liegt nicht im richtigen Format vor, um vom Integrationsendpunkt verwendet zu werden. Daher führt API Gateway eine Zuweisungsvorlagen-Transformation durch. API Gateway führt nur eine Transformation der Zuordnungsvorlage durch, weil für die eine Zuordnungsvorlage definiert ist Content-Type `application/json`. Wenn Sie keine Zuordnungsvorlage für das definieren Content-Type, leitet API Gateway den Hauptteil standardmäßig über die Integrationsanforderung an den Integrationsendpunkt weiter. Informationen zum Ändern dieses Verhaltens finden Sie unter [the section called “Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen”](#).

Die folgende Zuweisungsvorlage transformiert die Daten der Methodenanforderung in der Integrationsanforderung, bevor sie an den Integrationsendpunkt gesendet werden:

```
#set($inputRoot = $input.path('$'))
{
  "dogId" : "dog_"$elem.id,
  "Age": $inputRoot.Age
}
```

1. Die Variable `$inputRoot` stellt in den ursprünglichen JSON-Daten aus dem vorherigen Abschnitt das Stammobjekt dar. Richtlinien beginnen mit dem #-Symbol.
2. `dog` ist eine Verkettung der `id` des Benutzers und einer Zeichenfolge.
3. `Age` stammt aus dem Body der Methodenanforderung.

Anschließend wird die folgende Ausgabe an den Integrationsendpunkt weitergeleitet:

```
{
  "dogId" : "dog_1",
  "Age": 11
}
```

Integrationsantwort und Methodenantwort

Nach einer erfolgreichen Anfrage an den Integrationsendpunkt sendet der Endpunkt eine Antwort an die Integrationsantwort von API Gateway. Im Folgenden sehen Sie die Beispieldaten, die vom Integrationsendpunkt zurückgegeben werden:

```
{
  "dogId" : "dog_1",
  "adoptionFee": 19.95,
}
```

Die Methodenantwort erwartet andere Nutzdaten als die, die von der Integrationsantwort zurückgegeben werden. Daher führt API Gateway eine Transformation mit Zuweisungsvorlagen aus. API Gateway führt nur eine Transformation der Zuordnungsvorlage durch, weil für die eine Zuordnungsvorlage definiert ist Content-Type `application/json`. Wenn Sie keine Zuordnungsvorlage für die definieren Content-Type, leitet API Gateway standardmäßig den Hauptteil durch die Integrationsantwort an die Methodenantwort weiter. Informationen zum Ändern dieses Verhaltens finden Sie unter [the section called “Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen”](#).

```
#set($inputRoot = $input.path('$'))
{
  "adoptionFee" : $inputRoot.adoptionFee,
}
```

Die folgende Ausgabe wird an die Methodenantwort gesendet:

```
{"adoptionFee": 19.95}
```

Damit ist das Beispiel zur Transformation mit Zuweisungsvorlagen abgeschlossen. Wir empfehlen, wann immer möglich, statt Transformationen mit Zuweisungsvorlagen eine Proxy-Integration zu verwenden, um Ihre Daten zu transformieren. Weitere Informationen finden Sie unter [the section called “Auswahl eines API-Integrationstyps”](#).

Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway

Wenn Ihre Methodenanforderung Nutzdaten enthält und Sie keine Zuweisungsvorlage für den Content-Type-Header definiert haben, können Sie wählen, ob die vom Client bereitgestellten

Anfragenutzdaten ohne Transformation über die Integrationsanforderung an das Backend weitergegeben werden. Dieser Prozess wird als "Integrations-Pass-Through" bezeichnet.

Das tatsächliche Passthrough-Verhalten einer eingehenden Anfrage wird durch diese Einstellung bestimmt. Es gibt drei Optionen:

Wenn keine Vorlage mit dem Content-Type Anforderungsheader übereinstimmt

Wählen Sie diese Option, falls der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern der Inhaltstyp der Methodenanforderung keinem mit den Zuweisungsvorlagen verknüpften Inhaltstyp entspricht.

Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_MATCH` als `passthroughBehavior`-Eigenschaftswert für die [Integration](#) festlegen.

Wenn keine Vorlagen definiert sind (empfohlen)

Wählen Sie diese Option, falls der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern in der Integrationsanforderung keine Zuweisungsvorlage definiert ist. Wenn eine Vorlage definiert und diese Option ausgewählt ist, wird die Methodenanforderung mit Nutzdaten und einem Inhaltstyp, der keiner definierten Zuweisungsvorlage entspricht, mit einer HTTP-415-Antwort (Nicht unterstützter Medientype) zurückgewiesen.

Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_TEMPLATES` als `passthroughBehavior`-Eigenschaftswert für die [Integration](#) festlegen.

Niemals

Wählen Sie diese Option, falls der Methodenanforderungstext nicht ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern in der Integrationsanforderung keine Zuweisungsvorlage definiert ist. Wenn eine Vorlage definiert und diese Option ausgewählt ist, wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit einer Antwort "HTTP 415 Unsupported Media Type" zurückgewiesen.

Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `NEVER` als `passthroughBehavior`-Eigenschaftswert für die [Integration](#) festlegen.

Die folgenden Beispiele veranschaulichen die möglichen Pass-Through-Verhalten.

Beispiel 1: Eine Mapping-Vorlage wird in der Integrationsanforderung für den Content-Type `application/json` definiert.

Content-type	Pass-Through-Option	Behavior
Keine API Gateway verwendet standardmäßig applicati on/json	WHEN_NO_MATCH	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
Keine API Gateway verwendet standardmäßig applicati on/json	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
Keine API Gateway verwendet standardmäßig applicati on/json	NEVER	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/json	WHEN_NO_MATCH	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/json	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/json	NEVER	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/xml	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
application/xml	WHEN_NO_TEMPLATES	Die Anforderung wird mit der HTTP-Antwort 415

Content-type	Pass-Through-Option	Behavior
		Unsupported Media Type abgelehnt.
application/xml	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.

Beispiel 2: Eine Mapping-Vorlage wird in der Integrationsanforderung für den Content-Type `application/xml` definiert.

Content-type	Pass-Through-Option	Behavior
Keine API Gateway verwendet standardmäßig <code>application/json</code>	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
Keine API Gateway verwendet standardmäßig <code>application/json</code>	WHEN_NO_TEMPLATES	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
Keine API Gateway verwendet standardmäßig <code>application/json</code>	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
application/json	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.

Content-type	Pass-Through-Option	Behavior
application/json	WHEN_NO_TEMPLATES	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
application/json	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
application/xml	WHEN_NO_MATCH	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/xml	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
application/xml	NEVER	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.

Beispiel 3: In der Integrationsanforderung sind keine Zuweisungsvorlagen definiert.

Content-type	Pass-Through-Option	Behavior
Keine API Gateway verwendet standardmäßig applicati on/json	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
Keine	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.

Content-type	Pass-Through-Option	Behavior
API Gateway verwendet standardmäßig application/json		
Keine API Gateway verwendet standardmäßig application/json	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
application/json	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
application/json	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
application/json	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.
application/xml	WHEN_NO_MATCH	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.
application/xml	WHEN_NO_TEMPLATES	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.

Content-type	Pass-Through-Option	Behavior
application/xml	NEVER	Die Anforderung wird mit der HTTP-Antwort 415 Unsupported Media Type abgelehnt.

Zusätzliches Beispiel für Zuweisungsvorlagen für REST-APIs in API Gateway

Das folgende Beispiel zeigt eine Fotoalbum-API in API Gateway, die Zuweisungsvorlagen verwendet, um Daten der Integrationsanforderung und Integrationsantwort zu transformieren. Es werden außerdem Datenmodelle verwendet, um die Nutzdaten der Methodenanforderung und der Integrationsantwort zu definieren. Weitere Informationen zu Datenmodellen finden Sie unter [Datenmodelle für REST APIs](#).

Methodenanforderung und Integrationsanforderung

Im Folgenden ist ein Modell dargestellt, das den Body der Methodenanforderung definiert. Bei diesem Eingabemodell muss der Aufrufer mindestens eine Fotoseite hochladen und für jede Seite wird ein Minimum von 10 Fotos vorgegeben. Sie können dieses Eingabemodell verwenden, um ein SDK zu generieren oder um eine Anfragevalidierung für Ihre API zu nutzen. Bei der Verwendung der Anfragevalidierung lässt API Gateway die Anfrage fehlschlagen, wenn der Body der Methodenanforderung nicht der Datenstruktur des Modells entspricht.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosInputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "object",
      "required" : [
        "photo"
      ],
      "properties": {
        "page": { "type": "integer" },
        "pages": { "type": "string" },
        "perpage": { "type": "integer", "minimum" : 10 },
        "total": { "type": "string" },
        "photo": {
```



```

    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "id": { "type": "string" },
        "owner": { "type": "string" },
        "photographer_first_name" : {"type" : "string"},
        "photographer_last_name" : {"type" : "string"},
        "secret": { "type": "string" },
        "server": { "type": "string" },
        "farm": { "type": "integer" },
        "title": { "type": "string" },
        "ispublic": { "type": "boolean" },
        "isfriend": { "type": "boolean" },
        "isfamily": { "type": "boolean" }
      }
    }
  }
}
}
}
}
}

```

Im Folgenden sehen Sie ein Beispiel für einen Body der Methodenanforderung, der der Datenstruktur des vorherigen Datenmodells entspricht.

```

{
  "photos": {
    "page": 1,
    "pages": "1234",
    "perpage": 100,
    "total": "123398",
    "photo": [
      {
        "id": "12345678901",
        "owner": "23456789@A12",
        "photographer_first_name" : "Saanvi",
        "photographer_last_name" : "Sarkar",
        "secret": "abc123d456",
        "server": "1234",
        "farm": 1,
        "title": "Sample photo 1",
        "ispublic": true,

```

```

        "isfriend": false,
        "isfamily": false
    },
    {
        "id": "23456789012",
        "owner": "34567890@B23",
        "photographer_first_name" : "Richard",
        "photographer_last_name" : "Roe",
        "secret": "bcd234e567",
        "server": "2345",
        "farm": 2,
        "title": "Sample photo 2",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
    }
]
}
}

```

In diesem Beispiel transformiert die Zuweisungsvorlage die Nutzdaten der vorherigen Methodenanforderung, wenn sie vom Client übermittelt wurden, sodass das Format den Anforderungen des Integrationsendpunkts entspricht.

```

#set($inputRoot = $input.path('$'))
{
    "photos": [
#foreach($elem in $inputRoot.photos.photo)
        {
            "id": "$elem.id",
            "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
            "title": "$elem.title",
            "ispublic": $elem.ispublic,
            "isfriend": $elem.isfriend,
            "isfamily": $elem.isfamily
        }#if($foreach.hasNext),#end
#end
    ]
}

```

Im folgenden Beispiel sehen Sie die Ausgabedaten aus der Transformation.

```
{
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    }
  ]
}
```

Diese Daten werden an die Integrationsanforderung und anschließend an den Integrationsendpunkt gesendet.

Integrationsantwort und Methodenantwort

Im Folgenden sehen Sie ein Beispielausgabemodell für die Fotodaten vom Integrationsendpunkt. Sie können dieses Modell für ein Methodenantwortmodell verwenden; dies ist erforderlich, wenn Sie ein stark typisiertes SDK für die API generieren. Dies führt dazu, dass die Ausgabe in eine entsprechende Klasse in Java oder umgewandelt wird Objective-C.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },

```

```

        "photographedBy": { "type": "string" },
        "title": { "type": "string" },
        "ispublic": { "type": "boolean" },
        "isfriend": { "type": "boolean" },
        "isfamily": { "type": "boolean" }
    }
}
}
}
}

```

Der Integrationsendpunkt liefert möglicherweise keine Antwort, die der Datenstruktur dieses Modells entspricht. Zum Beispiel könnte die Integrationsantwort wie folgt aussehen:

```

"photos": [
{
    "id": "12345678901",
    "photographedBy": "Saanvi Sarkar",
    "title": "Sample photo 1",
    "description": "My sample photo 1",
    "public": true,
    "friend": false,
    "family": false
},
{
    "id": "23456789012",
    "photographedBy": "Richard Roe",
    "title": "Sample photo 2",
    "description": "My sample photo 1",
    "public": true,
    "friend": false,
    "family": false
}
]
}

```

Die folgende Zuweisungsvorlage transformiert die Integrationsantwortdaten in das Format, das von der Methodenantwort erwartet wird:

```

#set($inputRoot = $input.path('$'))
{

```

```
"photos": [  
  #foreach($elem in $inputRoot.photos.photo)  
    {  
      "id": "$elem.id",  
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",  
      "title": "$elem.title",  
      "ispublic": $elem.public,  
      "isfriend": $elem.friend,  
      "isfamily": $elem.family  
    }#if($foreach.hasNext),#end  
  #end  
]
```

Im folgenden Beispiel sehen Sie die Ausgabedaten aus der Transformation.

```
{  
  "photos": [  
    {  
      "id": "12345678901",  
      "photographedBy": "Saanvi Sarkar",  
      "title": "Sample photo 1",  
      "ispublic": true,  
      "isfriend": false,  
      "isfamily": false  
    },  
    {  
      "id": "23456789012",  
      "photographedBy": "Richard Roe",  
      "title": "Sample photo 2",  
      "ispublic": true,  
      "isfriend": false,  
      "isfamily": false  
    }  
  ]  
}
```

Diese Daten werden an die Methodenantwort gesendet und anschließend an den Client zurückgegeben.

Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway

Sie können Transformationen mit Zuweisungsvorlagen verwenden, um beliebige Arten von Anfrageparametern, Antwort-Headern oder Antwortstatuscodes zu überschreiben. Mit einer Zuweisungsvorlage können Sie Folgendes tun:

- Die Ausführung von Viele-zu-Einem-Parameterzuweisungen
- Eine Überschreibung von Parametern, nachdem standardmäßige API-Gateway-Zuweisungen angewendet wurden
- Die Zuordnung konditionaler Parameter auf Grundlage des Textinhalts oder anderer Parameterwerte
- Die programmgesteuerte Erstellung neuer Parameter
- Das Überschreiben von Statuscodes, die von Ihrem Integrationsendpunkt zurückgegeben wurden

Überschreibungen sind endgültig. Eine Überschreibung darf nur einmal auf jeden Parameter angewendet werden. Wenn Sie versuchen, denselben Parameter mehrmals zu überschreiben, gibt API Gateway eine 5XX-Antwort zurück. Wenn Sie denselben Parameter in einer Vorlage mehrmals überschreiben müssen, empfehlen wir die Erstellung einer Variable und Umsetzung der Überschreibung am Ende der Vorlage. Die Vorlage wird erst nach der Analyse der gesamten Vorlage angewendet.

Beispiel 1: Überschreiben des Statuscodes basierend auf dem Body der Integration

Im folgenden Beispiel wird die [Beispiel-API](#) verwendet, um den Statuscode basierend auf dem Body der Integrationsantwort zu überschreiben.

AWS-Managementkonsole

Überschreiben eines Statuscodes basierend auf dem Body der Integrationsantwort

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie Create API (API erstellen) aus.
3. Wählen Sie unter REST-API die Option Erstellen aus.
4. Wählen Sie unter API-Details die Option Beispiel-API aus.
5. Wählen Sie Create API (API erstellen) aus.

API Gateway erstellt eine Beispiel-API für einen PetStore. Um Informationen über ein Haustier abzurufen, verwenden Sie die API-Methodenanforderung `GET /pets/{petId}`, wobei `{petId}` ein Pfadparameter ist, der einer ID-Nummer für ein Haustier entspricht.

In diesem Beispiel überschreiben Sie den Antwortcode der GET-Methode auf `400`, wenn eine Fehlerbedingung erkannt wird.

6. Wählen Sie in der Ressourcen-Struktur unter GET die `/petId`-Methode aus.
7. Testen Sie zunächst die aktuelle Implementierung der API.

Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

8. Geben Sie für `petId` `-1` ein und klicken Sie dann auf Test.

Der Antwort-Body zeigt einen Fehler außerhalb des gültigen Bereichs an:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Außerdem endet die letzte Zeile unter Logs mit: `Method completed with status: 200`.

Die Integration wurde erfolgreich abgeschlossen, es ist jedoch ein Fehler aufgetreten. Jetzt überschreiben Sie den Statuscode auf Grundlage der Integrationsantwort.

9. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
10. Wählen Sie Zuordnungsvorlagen aus.
11. Wählen Sie Add mapping template.
12. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
13. Geben Sie für Vorlagentext Folgendes ein:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
```

```
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

Diese Zuweisungsvorlage verwendet die Variable `$context.responseOverride.status`, um den Statuscode auf 400 zu überschreiben, wenn die Integrationsantwort den Zeichenfolge `error` enthält.

14. Wählen Sie Speichern.
15. Wählen Sie die Registerkarte Test.
16. Geben Sie als `petid` **-1** ein.
17. In den Ergebnissen zeigt der Antworttext einen Fehler „außerhalb des gültigen Bereichs“ an:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Die letzte Zeile unter dem Logs endet nun jedoch mit: `Method completed with status: 400`.

CloudFormation

In diesem Beispiel verwenden Sie die [body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 1
          description: Example pet store API.
```



```

    version: "2025-01-14T00:13:18Z"
  paths:
    /pets/{petId}:
      get:
        parameters:
          - name: petId
            in: path
            required: true
            schema:
              type: string
        responses:
          "200":
            description: 200 response
        x-amazon-apigateway-integration:
          httpMethod: GET
          uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}
        responses:
          default:
            statusCode: "200"
            responseTemplates:
              application/json: |-
                #set($inputRoot = $input.path('$'))
                $input.json("$")
                #if($inputRoot.toString().contains("error"))
                #set($context.responseOverride.status = 400)
                #end
            requestParameters:
              integration.request.path.petId: method.request.path.petId
          passthroughBehavior: when_no_match
          type: http
  components:
    schemas:
      Pet:
        type: object
        properties:
          id:
            type: integer
          type:
            type: string
          price:
            type: number
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'

```

```

    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      DeploymentId: !Ref ApiGatewayDeployment20250219
      RestApiId: !Ref Api
      StageName: prod

```

OpenAPI

Die folgende OpenAPI-Definition erstellt die GET `pets/{petId}`-Ressource und überschreibt den Statuscode basierend auf dem Body der Integration.

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 1",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:13:18Z"
  },
  "paths" : {
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        }
      }
    }
  },

```

```

    "x-amazon-apigateway-integration" : {
      "httpMethod" : "GET",
      "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}",
      "responses" : {
        "default" : {
          "statusCode" : "200",
          "responseTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n
$input.json(\"$\")\n#if($inputRoot.toString().contains(\"error
\"))\n#set($context.responseOverride.status = 400)\n#end"
          }
        }
      },
      "requestParameters" : {
        "integration.request.path.petId" : "method.request.path.petId"
      },
      "passthroughBehavior" : "when_no_match",
      "type" : "http"
    }
  }
},
"components" : {
  "schemas" : {
    "Pet" : {
      "type" : "object",
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string"
        },
        "price" : {
          "type" : "number"
        }
      }
    }
  }
}
}
}

```

Beispiel 2: Überschreiben des Anfrage-Headers und Erstellen neuer Header

Das folgende Beispiel verwendet die [Beispiel-API](#), um den Anfrage-Header zu überschreiben und neue Header zu erstellen.

AWS-Managementkonsole

So überschreiben Sie einen Methodenanforderungs-Header indem Sie einen neuen Header erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter an <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie die Beispiel-API aus, die Sie im vorherigen Tutorial erstellt haben. Der Name der API sollte lauten PetStore.
3. Wählen Sie in der Ressourcen-Struktur unter GET die /pet-Methode aus.
4. Klicken Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen auf Bearbeiten.
5. Wählen Sie HTTP Request Headers (HTTP-Anforderungs-Header) und dann Add header (Header hinzufügen) aus.
6. Geben Sie unter Name **header1** ein.
7. Klicken Sie auf Header hinzufügen und erstellen Sie dann einen zweiten Header namens **header2**.
8. Wählen Sie Speichern aus.

Kombinieren Sie jetzt diese Header mithilfe einer Zuweisungsvorlage zu einem einzigen Header-Wert.

9. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
10. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.
11. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Add mapping template.
 - b. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
 - c. Geben Sie für Vorlagentext Folgendes ein:

```
#set($header10override = "pets")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.request0override.header.header3 = $header3Value)
#set($context.request0override.header.header1 = $header10override)
#set($context.request0override.header.multivalueheader=[$header10override,
$header3Value])
```

Diese Zuweisungsvorlage überschreibt `header1` mit der Zeichenfolge `pets` und erstellt einen mehrwertigen Header mit dem Namen `$header3Value`, der `header1` und `header2` kombiniert.

12. Wählen Sie Speichern.
13. Wählen Sie die Registerkarte Test.
14. Kopieren Sie unter Headers den folgenden Code:

```
header1:header1Val
header2:header2Val
```

15. Wählen Sie Test aus.

In den Logs sollten Sie einen Eintrag sehen, der diesen Text enthält:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=pets, x-amzn-apigateway-api-id=api-id,
Accept=application/json, multivalueheader=pets,header1Valheader2Val}
```

CloudFormation

In diesem Beispiel verwenden Sie die [body](#)-Eigenschaft, um eine OpenAPI-Definitionsdatei in API Gateway zu importieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
```

```

info:
  title: PetStore Example 2
  description: Example pet store API.
  version: "2025-01-14T00:36:18Z"
paths:
  /pets:
    get:
      parameters:
        - name: header2
          in: header
          schema:
            type: string
        - name: page
          in: query
          schema:
            type: string
        - name: type
          in: query
          schema:
            type: string
        - name: header1
          in: header
          schema:
            type: string
      responses:
        "200":
          description: 200 response
      x-amazon-apigateway-integration:
        httpMethod: GET
        uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/

pets
  responses:
    default:
      statusCode: "200"
  requestParameters:
    integration.request.header.header1: method.request.header.header1
    integration.request.header.header2: method.request.header.header2
    integration.request.querystring.page:
method.request.querystring.page
    integration.request.querystring.type:
method.request.querystring.type
  requestTemplates:
    application/json: |-
      #set($header10override = "pets")

```

```

        #set($header3Value =
"$input.params('header1')$input.params('header2')")
        $input.json("$")
        #set($context.requestOverride.header.header3 = $header3Value)
        #set($context.requestOverride.header.header1 = $header10override)

#set($context.requestOverride.header.multivalueheader=[$header10override,
$header3Value])
        passthroughBehavior: when_no_match
        type: http
    components:
        schemas:
            Pet:
                type: object
                properties:
                    id:
                        type: integer
                    type:
                        type: string
                    price:
                        type: number
    ApiGatewayDeployment:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    ApiGatewayDeployment20250219:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    Stage:
        Type: 'AWS::ApiGateway::Stage'
        Properties:
            DeploymentId: !Ref ApiGatewayDeployment20250219
            RestApiId: !Ref Api
            StageName: prod

```

OpenAPI

Die folgende OpenAPI-Definition erstellt die GET `pets`-Ressource, überschreibt den Anfrage-Header und erstellt neue Header.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 2",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:36:18Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "header2",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "page",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "type",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "header1",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }
      ],
      "responses" : {
        "200" : {
          "description" : "200 response"
        }
      },
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
```



```

    "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets",
    "responses" : {
        "default" : {
            "statusCode" : "200"
        }
    },
    "requestParameters" : {
        "integration.request.header.header1" : "method.request.header.header1",
        "integration.request.header.header2" : "method.request.header.header2",
        "integration.request.querystring.page" :
"method.request.querystring.page",
        "integration.request.querystring.type" :
"method.request.querystring.type"
    },
    "requestTemplates" : {
        "application/json" : "#set($header10override = \"pets
\")\n#set($header3Value = \"$input.params('header1')$input.params('header2')\")\n
$input.json(\"$\")\n#set($context.request0override.header.header3
= $header3Value)\n#set($context.request0override.header.header1 =
$header10override)\n#set($context.request0override.header.multivalueheader=[
$header10override,
$header3Value])"
    },
    "passthroughBehavior" : "when_no_match",
    "type" : "http"
}
}
}
}
}
}
}

```

Um eine Zuweisungsvorlage zum Überschreiben zu verwenden, fügen Sie eine oder mehrere der folgenden `$context`-Variablen hinzu. Sie finden eine Liste der `$context`-Variablen unter [the section called “Kontextvariablen für Datentransformationen”](#).

Tutorial: Ändern Sie die Integrationsanfrage und die Antwort für Integrationen zu AWS service

Das folgende Tutorial zeigt, wie Sie mithilfe von Mapping-Vorlagentransformationen Mapping-Vorlagen einrichten, um Integrationsanfragen und -antworten mithilfe der Konsole und der AWS CLI zu transformieren.

Themen

- [Einrichten einer Datentransformation mithilfe der API-Gateway-Konsole](#)
- [Richten Sie die Datentransformation mit dem ein AWS CLI](#)
- [Datentransformation abgeschlossen CloudFormation Vorlage](#)

Einrichten einer Datentransformation mithilfe der API-Gateway-Konsole

In diesem Tutorial erstellen Sie eine unvollständige API- und DynamoDB-Tabelle mithilfe der folgenden ZIP-Datei [data-transformation-tutorial-console.zip](#). Diese unvollständige API hat eine /pets-Ressource mit den Methoden GET und POST.

- Die GET-Methode ruft Daten vom `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt ab. Die Ausgabedaten werden gemäß der Zuweisungsvorlage zu [the section called "Transformationen mit Zuweisungsvorlagen"](#) transformiert.
- Die POST-Methode ermöglicht dem Benutzer die POST-Aktion von Haustierinformationen mithilfe einer Zuweisungsvorlage an eine Amazon-DynamoDB-Tabelle.

Laden Sie [die Vorlage zur App-Erstellung](#) für herunter und entpacken Sie sie. CloudFormation Sie verwenden diese Vorlage, um eine DynamoDB-Tabelle zum Posten von Haustierinformationen und einer unvollständigen API zu erstellen. Sie schließen die restlichen Schritte in der API-Gateway-Konsole ab.

Um eine zu erstellen CloudFormation Stack

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.
6. Geben Sie für Stack-Name die Zeichenfolge **data-transformation-tutorial-console** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden CloudFormation können.
9. Wählen Sie Weiter und anschließend Absenden aus.

CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres CloudFormation Stacks `CREATE_COMPLETE` lautet, können Sie mit dem nächsten Schritt fortfahren.

Um die Antwort der **GET-Integration** zu testen

1. Wählen Sie auf der Registerkarte Ressourcen des CloudFormation Stacks für **data-transformation-tutorial-console** die physische ID Ihrer API aus.
2. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die GET-Methode in der Ressourcenstruktur aus.
3. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

Das Ergebnis des Tests wird Folgendes zeigen:

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Sie transformieren diese Ausgabe gemäß der Zuweisungsvorlage in [the section called “Transformationen mit Zuweisungsvorlagen”](#).

Um die GET-Integrationsantwort **zu** transformieren

1. Klicken Sie auf die Registerkarte Integrationsantwort.

Derzeit sind keine Zuweisungsvorlagen definiert, sodass die Integrationsantwort nicht transformiert wird.

2. Klicken Sie unter Standard - Antwort auf Bearbeiten.
3. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:
 - a. Wählen Sie Add mapping template.
 - b. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
 - c. Geben Sie für Vorlagentext Folgendes ein:

```
#set($inputRoot = $input.path('$'))
[
  #foreach($elem in $inputRoot)
    {
      "description" : "Item $elem.id is a $elem.type.",
      "askingPrice" : $elem.price
    }#if($foreach.hasNext),#end
  #end
]
```

Wählen Sie Speichern.

Um die **GET-Integrationsantwort** zu testen

- Wählen Sie die Registerkarte Test und dann Test aus

Die Ausgabe des Tests zeigt die transformierte Antwort.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
```

```
"description" : "Item 3 is a fish.",  
"askingPrice" : 0.99  
}  
]
```

Um Eingabedaten aus der **POST-Methode** zu transformieren

1. Wählen Sie die POST-Methode aus.
2. Wählen Sie die Registerkarte Integrationsanfrage aus und klicken Sie dann unter Einstellungen für Integrationsanforderungen auf Bearbeiten.

Die CloudFormation Vorlage hat einige Felder für Integrationsanfragen gefüllt.

- Der Integrationstyp ist AWS-Service.
- Das AWS-Service ist DynamoDB.
- Die HTTP-Methode ist POST.
- Die Aktion ist PutItem.
- Die Ausführungsrolle, die es API Gateway ermöglicht, ein Element in die DynamoDB-Tabelle aufzunehmen, lautet. `data-transformation-tutorial-console-APIGatewayRole` CloudFormation hat diese Rolle erstellt, damit API Gateway über die Mindestberechtigungen für die Interaktion mit DynamoDB verfügt.

Der Name der DynamoDB-Tabelle wurde nicht angegeben. In den folgenden Schritten geben Sie den Namen an.

3. Für Text-Passthrough anfordern wählen Sie Nie aus.

Das bedeutet, dass die API Daten ablehnt, für Content-Types die es keine Zuordnungsvorlage gibt.

4. Wählen Sie Zuordnungsvorlagen aus.
5. Der Inhaltstyp ist auf `application/json` gesetzt. Das bedeutet, dass Inhaltstypen, die `application/json` dies nicht sind, von der API abgelehnt werden. Weitere Informationen zu Integrations-Passthrough-Verhaltensweisen finden Sie unter [Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway](#).
6. Geben Sie den folgenden Code in den Texteditor ein.

```
{
```

```
"TableName": "data-transformation-tutorial-console-ddb",
"Item": {
  "id": {
    "N": $input.json("$.id")
  },
  "type": {
    "S": $input.json("$.type")
  },
  "price": {
    "N": $input.json("$.price")
  }
}
```

Diese Vorlage spezifiziert die Tabelle als `data-transformation-tutorial-console-ddb` und legt die Elemente als `id`, `type` und `price` fest. Die Elemente kommen aus dem Hauptteil der **POST**-Methode. Sie können auch ein Datenmodell verwenden, um eine Zuweisungsvorlage zu erstellen. Weitere Informationen finden Sie unter [Validierung für REST APIs in API Gateway anfordern](#).

7. Betätigen Sie die Schaltfläche **Speichern**, um die Zuweisungsvorlage zu speichern.

Um eine Methode und eine Integrationsantwort aus der **POST-Methode** hinzuzufügen

Sie CloudFormation haben eine leere Methode und eine leere Integrationsantwort erstellt. Sie werden diese Antwort bearbeiten, um weitere Informationen bereitzustellen. Weitere Informationen zum Bearbeiten von Antworten finden Sie unter [Beispiele für Parameterzuweisungen für REST-APIs in API Gateway](#).

1. Klicken Sie auf der Registerkarte Integrationsantwort unter **Standard - Antwort auf Bearbeiten**.
2. Wählen Sie Zuordnungsvorlagen aus und klicken Sie dann auf **Zuordnungsvorlage hinzufügen**.
3. Geben Sie unter **Content-type** den Wert **application/json** ein.
4. Geben Sie im Code-Editor die folgende Ausgabe–Zuweisungsvorlage ein, um eine Ausgabenachricht zu senden:

```
{ "message" : "Your response was recorded at $context.requestTime" }
```

Weitere Informationen zu Kontextvariablen finden Sie unter [Kontextvariablen für Datentransformationen](#).

5. Betätigen Sie die Schaltfläche Speichern, um die Zuweisungsvorlage zu speichern.

Testen Sie die **POST-Methode**

Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

1. Geben Sie im Anforderungstext das folgende Beispiel ein.

```
{
    "id": "4",
    "type" : "dog",
    "price": "321"
}
```

2. Wählen Sie Test aus.

Die Ausgabe sollte Ihre Erfolgsmeldung enthalten.

Sie können die DynamoDB-Konsole unter öffnen, <https://console.aws.amazon.com/dynamodb/>um zu überprüfen, ob sich das Beispielement in Ihrer Tabelle befindet.

Um ein zu löschen CloudFormation Stack

1. Öffnen Sie die CloudFormation Konsole unter <https://console.aws.amazon.com/cloudformation>.
2. Wählen Sie Ihren CloudFormation Stack aus.
3. Wählen Sie Löschen und bestätigen Sie dann Ihre Auswahl.

Richten Sie die Datentransformation mit dem ein AWS CLI

In diesem Tutorial erstellen Sie eine unvollständige API- und DynamoDB-Tabelle mithilfe der folgenden ZIP-Datei [data-transformation-tutorial-cli.zip](#). Diese unvollständige API verfügt über eine /pets-Ressource mit einer GET-Methode, die in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt integriert ist. Sie erstellen eine POST-Methode, um eine Verbindung zu einer DynamoDB-Tabelle herzustellen, und verwenden Zuweisungsvorlagen, um Daten in eine DynamoDB-Tabelle einzugeben.

- Sie transformieren die Ausgabedaten gemäß der Zuweisungsvorlage zu [the section called "Transformationen mit Zuweisungsvorlagen"](#).

- Sie erstellen eine POST-Methode, die es dem Benutzer ermöglicht, mithilfe einer Zuweisungsvorlage die POST-Aktion an eine Amazon-DynamoDB-Tabelle durchzuführen.

Um eine zu erstellen CloudFormation Stack

Laden Sie [die Vorlage zur App-Erstellung für CloudFormation](#) herunter und entpacken Sie sie.

Um die folgenden Schritte durchzuführen, benötigen Sie die [AWS Command Line Interface \(AWS CLI\) Version 2](#).

Bei langen Befehlen wird ein Escape-Zeichen (\) verwendet, um einen Befehl auf mehrere Zeilen aufzuteilen.

 Note

In Windows werden einige Bash-CLI-Befehle, die Sie häufig verwenden (z. B. zip), von den integrierten Terminals des Betriebssystems nicht unterstützt. Um eine Windows-integrierte Version von Ubuntu und Bash zu erhalten, [installieren Sie das Windows-Subsystem](#) für Linux. Die CLI-Beispielbefehle in diesem Handbuch verwenden die Linux-Formatierung. Befehle, die Inline-JSON-Dokumente enthalten, müssen neu formatiert werden, wenn Sie die Windows-CLI verwenden.

1. Verwenden Sie den folgenden Befehl, um den Stack zu erstellen. CloudFormation

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
--template-body file://data-transformation-tutorial-cli.zip --capabilities
CAPABILITY_NAMED_IAM
```

2. CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Verwenden Sie den folgenden Befehl, um den Status Ihres CloudFormation Stacks zu überprüfen.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Wenn der Status Ihres CloudFormation Stacks lautet `StackStatus: "CREATE_COMPLETE"`, verwenden Sie den folgenden Befehl, um relevante Ausgabewerte für future Schritte abzurufen.


```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli  
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,  
Description: Description}"
```

Die Ausgabewerte sind die folgenden:

- **ApiRole**, das ist der Rollename, der es API Gateway ermöglicht, Elemente in die DynamoDB-Tabelle einzufügen. Für dieses Tutorial ist der Rollename `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
- **DDBTableName**, das ist der Name der DynamoDB-Tabelle. Für dieses Tutorial ist der Name der Tabelle `data-transformation-tutorial-cli-ddb`.
- **ResourceId**, das ist die ID der Pets-Ressource, in der die POST Methoden GET und verfügbar gemacht werden. Für dieses Tutorial lautet die Ressourcen-ID `efg456`.
- **ApilId**, was die ID für die API ist. Für dieses Tutorial lautet die API-ID `abc123`.

Um die **GET-Methode** vor der Datentransformation zu testen

- Geben Sie den folgenden Befehl ein, um die GET-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method GET
```

Die Ausgabe des Tests wird Folgendes zeigen.

```
[  
  {  
    "id": 1,  
    "type": "dog",  
    "price": 249.99  
  },  
  {  
    "id": 2,  
    "type": "cat",  
    "price": 124.99  
  },  
  {  
    "id": 3,
```

```
    "type": "fish",
    "price": 0.99
  }
]
```

Sie transformieren diese Ausgabe gemäß der Zuweisungsvorlage in [the section called “Transformationen mit Zuweisungsvorlagen”](#).

Um die GET-Integrationsantwort **zu** transformieren

- Führen Sie den folgenden Befehl aus, um die Integrationsantwort für die GET-Methode zu aktualisieren. Ersetzen Sie das *rest-api-id* und *resource-id* durch Ihre Werte.

Verwenden Sie den folgenden Befehl, um die Integrationsantwort zu erstellen.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "#set($inputRoot = $input.path(\"$\n\")).\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a\n  $elem.type\", \n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n\n#end\n]"]' }
```

Um die GET-Methode **zu** testen

- Geben Sie den folgenden Befehl ein, um die GET-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

Die Ausgabe des Tests zeigt die transformierte Antwort.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
]
```

```
{
  "description" : "Item 2 is a cat.",
  "askingPrice" : 124.99
},
{
  "description" : "Item 3 is a fish.",
  "askingPrice" : 0.99
}
]
```

Um eine **POST-Methode** zu erstellen

1. Führen Sie den folgenden Befehl aus, um eine neue Methode für die /pets-Ressource zu erstellen.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Mit dieser Methode können Sie Pet-Informationen an die DynamoDB-Tabelle senden, die Sie im Stack erstellt haben. CloudFormation

2. Verwenden Sie den folgenden Befehl, um eine AWS-Service Integration für die POST Methode zu erstellen.

```
aws apigateway put-integration --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --type AWS \
  --integration-http-method POST \
  --uri "arn:aws:apigateway:us-east-2:dynamodb:action/PutItem" \
  --credentials arn:aws:iam::111122223333:role/data-transformation-tutorial-cli-APIGatewayRole-ABCDEFGF \
  --request-templates '{"application/json":{"\TableName\":"data-transformation-tutorial-cli-ddb\","\Item\":{"id\":{"N\":"$input.json(\$.id\)"}},"type\":{"S\":"$input.json(\$.type\)"}},"price\":{"N\":"$input.json(\$.price\)"}} }' }
```

3. Verwenden Sie den folgenden Befehl, um eine Methodenantwort für einen erfolgreichen Aufruf der POST-Methode zu erstellen.

```
aws apigateway put-method-response --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --status-code 200
```

4. Verwenden Sie den folgenden Befehl, um eine Integrationsantwort für den erfolgreichen Aufruf der POST-Methode zu erstellen.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --status-code 200 \  
  --selection-pattern "" \  
  --response-templates '{"application/json": "{\\"message\\": \\"Your response was recorded at $context.requestTime\\"}"}'
```

Um die **POST-Methode** zu testen

- Geben Sie den folgenden Befehl ein, um die POST-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --body '{"id": "4", "type": "dog", "price": "321"}'
```

In der Ausgabe wird die Erfolgsmeldung angezeigt.

Um ein zu löschen CloudFormation Stack

- Verwenden Sie den folgenden Befehl, um Ihre CloudFormation Ressourcen zu löschen.

```
aws cloudformation delete-stack --stack-name data-transformation-tutorial-cli
```

Datentransformation abgeschlossen CloudFormation Vorlage

Das folgende Beispiel ist eine fertige CloudFormation Vorlage, die eine API und eine DynamoDB-Tabelle mit einer /pets Ressource mit GET und POST Methoden erstellt.

- Die GET-Methode ruft Daten vom `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt ab. Die Ausgabedaten werden gemäß der Zuweisungsvorlage zu [the section called “Transformationen mit Zuweisungsvorlagen”](#) transformiert.
- Die POST-Methode ermöglicht es dem Benutzer, mithilfe einer Zuweisungsvorlage die POST-Aktion zu einer DynamoDB-Tabelle durchzuführen.

Beispiel CloudFormation Vorlage

```
AWSTemplateFormatVersion: 2010-09-09
Description: A completed Amazon API Gateway REST API that uses non-proxy integration
  to POST to an Amazon DynamoDB table and non-proxy integration to GET transformed pets
  data.
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  DynamoDBTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      TableName: !Sub data-transformation-tutorial-complete
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: N
      KeySchema:
        - AttributeName: id
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
  APIGatewayRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
```

```

        - apigateway.amazonaws.com
Policies:
  - PolicyName: APIGatewayDynamoDBPolicy
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - 'dynamodb:PutItem'
          Resource: !GetAtt DynamoDBTable.Arn
Api:
  Type: 'AWS::ApiGateway::RestApi'
  Properties:
    Name: data-transformation-complete-api
    ApiKeySourceType: HEADER
PetsResource:
  Type: 'AWS::ApiGateway::Resource'
  Properties:
    RestApiId: !Ref Api
    ParentId: !GetAtt Api.RootResourceId
    PathPart: 'pets'
PetsMethodGet:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref PetsResource
    HttpMethod: GET
    ApiKeyRequired: false
    AuthorizationType: NONE
  Integration:
    Type: HTTP
    Credentials: !GetAtt APIGatewayRole.Arn
    IntegrationHttpMethod: GET
    Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
    PassthroughBehavior: WHEN_NO_TEMPLATES
    IntegrationResponses:
      - StatusCode: '200'
        ResponseTemplates:
          application/json: "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
$elem.type\", \n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"
    MethodResponses:
      - StatusCode: '200'

```

```

PetsMethodPost:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref PetsResource
    HttpMethod: POST
    ApiKeyRequired: false
    AuthorizationType: NONE
    Integration:
      Type: AWS
      Credentials: !GetAtt APIGatewayRole.Arn
      IntegrationHttpMethod: POST
      Uri: arn:aws:apigateway:us-west-1:dynamodb:action/PutItem
      PassthroughBehavior: NEVER
      RequestTemplates:
        application/json: "{\"TableName\":\"data-transformation-tutorial-complete
\", \"Item\": {\"id\": {\"N\": $input.json(\"$.id\")}, \"type\": {\"S\": $input.json(\"$.type
\")}, \"price\": {\"N\": $input.json(\"$.price\")} } }"
      IntegrationResponses:
        - StatusCode: 200
          ResponseTemplates:
            application/json: "{\"message\": \"Your response was recorded at
$context.requestTime\"}"
      MethodResponses:
        - StatusCode: '200'

ApiDeployment:
  Type: 'AWS::ApiGateway::Deployment'
  DependsOn:
    - PetsMethodGet
  Properties:
    RestApiId: !Ref Api
    StageName: !Sub '${StageName}'
Outputs:
  ApiId:
    Description: API ID for CLI commands
    Value: !Ref Api
  ResourceId:
    Description: /pets resource ID for CLI commands
    Value: !Ref PetsResource
  ApiRole:
    Description: Role ID to allow API Gateway to put and scan items in DynamoDB table
    Value: !Ref APIGatewayRole
  DDBTableName:

```

Description: DynamoDB table name
Value: !Ref DynamoDBTable

Beispiele für die Verwendung von Variablen bei Zuweisungsvorlagen-Transformationen in API Gateway

Die folgenden Beispiele zeigen, wie Sie `$context`-, `input`- und `util`-Variablen in Zuweisungsvorlagen verwenden. Sie können eine Scheinintegration oder eine Lambda-Nicht-Proxy-Integration verwenden, die das Eingabeereignis an API Gateway zurückgibt. Eine Liste aller unterstützten Variablen für Datentransformationen finden Sie unter [the section called “Variablen für Datentransformationen”](#).

Beispiel 1: Übergeben Sie mehrere **`$context`-Variablen** an den Integrationsendpunkt

Das folgende Beispiel zeigt eine Mapping-Vorlage für die Zuordnung eingehender `$context` Variablen zu Backend-Variablen mit geringfügig unterschiedlichen Namen in der Nutzlast einer Integrationsanforderung:

```
{
  "stage" : "$context.stage",
  "request_id" : "$context.requestId",
  "api_id" : "$context.apiId",
  "resource_path" : "$context.resourcePath",
  "resource_id" : "$context.resourceId",
  "http_method" : "$context.httpMethod",
  "source_ip" : "$context.identity.sourceIp",
  "user-agent" : "$context.identity.userAgent",
  "account_id" : "$context.identity.accountId",
  "api_key" : "$context.identity.apiKey",
  "caller" : "$context.identity.caller",
  "user" : "$context.identity.user",
  "user_arn" : "$context.identity.userArn"
}
```

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```
{
  stage: 'prod',
  request_id: 'abcdefg-000-000-0000-abcdefg',
  api_id: 'abcd1234',
  resource_path: '/',
}
```



```

resource_id: 'efg567',
http_method: 'GET',
source_ip: '192.0.2.1',
user-agent: 'curl/7.84.0',
account_id: '111122223333',
api_key: 'MyTestKey',
caller: 'ABCD-0000-12345',
user: 'ABCD-0000-12345',
user_arn: 'arn:aws:sts::111122223333:assumed-role/Admin/carlos-salazar'
}

```

Eine der Variablen ist ein API-Schlüssel. In diesem Beispiel wird vorausgesetzt, dass die Methode einen API-Schlüssel erfordert.

Beispiel 2: Übergeben aller Anfrageparameter an den Integrationsendpunkt über Nutzdaten im JSON-Format

Das folgende Beispiel übergibt alle Anfrageparameter, einschließlich path-, querystring- und header-Parameter, an den Integrationsendpunkt über Nutzdaten im JSON-Format:

```

#set($allParams = $input.params())
{
  "params" : {
    #foreach($type in $allParams.keySet())
    #set($params = $allParams.get($type))
    "$type" : {
      #foreach($paramName in $params.keySet())
      "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
      #if($foreach.hasNext),#end
      #end
    }
    #if($foreach.hasNext),#end
  }
}

```

Wenn eine Anfrage die folgenden Eingabeparameter enthält:

- Einen Pfadparameter namens myparam
- Die Abfragezeichenfolgenparameter querystring1=value1,value2
- Headers "header1" : "value1"

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```
{"params":{"path":{"example2":"myparamm"},"querystring":  
{"querystring1":"value1,value2"},"header":{"header1":"value1"}}}
```

Beispiel 3: Übergeben eines Unterabschnitts einer Methodenanforderung an den Integrationsendpunkt

Im folgenden Beispiel wird der Eingabeparameter `name` verwendet, um nur den Parameter `name` abzurufen, und der Eingabeparameter `input.json('$')`, um den gesamten Body der Methodenanforderung abzurufen:

```
{  
  "name" : "$input.params('name')",  
  "body" : $input.json('$')  
}
```

Für eine Anforderung, die die Parameter der Abfragezeichenfolge `name=Bella&type=dog` und den folgenden Text enthält:

```
{  
  "Price" : "249.99",  
  "Age": "6"  
}
```

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```
{  
  "name" : "Bella",  
  "body" : {"Price":"249.99","Age":"6"}  
}
```

Diese Zuweisungsvorlage entfernt den Abfragezeichenfolgenparameter `type=dog`.

Wenn die JSON-Eingabe Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können, gibt API Gateway möglicherweise eine 400-Antwort zurück. Eine Anwendung von `$util.escapeJavaScript($input.json('$'))` stellt sicher, dass die JSON-Eingabe ordnungsgemäß eingelesen werden kann.

Im vorangegangenen Beispiel hätte eine Anwendung von `$util.escapeJavaScript($input.json('$'))` folgendes Ergebnis:

```
{
  "name" : "$input.params('name')",
  "body" : "$util.escapeJavaScript($input.json('$'))"
}
```

In diesem Fall sollte die Ausgabe dieser Zuweisungsvorlage wie folgt aussehen:

```
{
  "name" : "Bella",
  "body": {"Price":"249.99","Age":"6"}
}
```

Beispiel 4: Verwenden eines JSONPath-Ausdrucks, um einen Unterabschnitt einer Methodenanforderung an den Integrationsendpunkt zu übergeben

Im folgenden Beispiel werden die JSONPath-Ausdrücke verwendet, um nur die Eingabeparameter name und Age aus dem Body der Anfrage abzurufen:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('$.Age')
}
```

Für eine Anforderung, die die Parameter der Abfragezeichenfolge name=Bella&type=dog und den folgenden Text enthält:

```
{
  "Price" : "249.99",
  "Age": "6"
}
```

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```
{
  "name" : "Bella",
  "body" : "6"
}
```

Diese Zuweisungsvorlage entfernt den Abfragezeichenfolgenparameter type=dog sowie das Feld Price aus dem Body.

Wenn die Payload einer Methodenanforderung Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können JavaScript, gibt API Gateway möglicherweise eine Antwort zurück. 400 Eine Anwendung von `$util.escapeJavaScript()` stellt sicher, dass die JSON-Eingabe ordnungsgemäß eingelesen werden kann.

Im vorangegangenen Beispiel hätte eine Anwendung von `$util.escapeJavaScript($input.json('$.Age'))` folgendes Ergebnis:

```
{
  "name" : "$input.params('name')",
  "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

In diesem Fall sollte die Ausgabe dieser Zuweisungsvorlage wie folgt aussehen:

```
{
  "name" : "Bella",
  "body": "\"6\""
}
```

Beispiel 5: Verwenden eines JSONPath-Ausdrucks, um Informationen über eine Methodenanforderung an den Integrationsendpunkt zu übergeben

Im folgenden Beispiel werden `$input.params()`, `$input.path()` und `$input.json()` verwendet, um Informationen über eine Methodenanforderung an den Integrationsendpunkt zu senden. Diese Zuweisungsvorlage nutzt die Methode `size()`, um die Anzahl der Elemente in einer Liste bereitzustellen.

```
{
  "id" : "$input.params('id')",
  "count" : "$input.path('$.things').size()",
  "things" : $input.json('$.things')
}
```

Für eine Anforderung, die den Pfadparameter 123 und den folgenden Text enthält:

```
{
  "things": {
    "1": {},
  },
}
```

```

        "2": {},
        "3": {}
    }
}

```

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```

{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}

```

Wenn die Payload einer Methodenanforderung Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können JavaScript, gibt API Gateway möglicherweise eine Antwort zurück. 400 Eine Anwendung von `$util.escapeJavaScript()` stellt sicher, dass die JSON-Eingabe ordnungsgemäß eingelesen werden kann.

Im vorangegangenen Beispiel hätte eine Anwendung von `$util.escapeJavaScript($input.json('$.things'))` folgendes Ergebnis:

```

{
  "id" : "$input.params('id')",
  "count" : "$input.path('$.things').size()",
  "things" : "$util.escapeJavaScript($input.json('$.things'))"
}

```

Die Ausgabe dieser Zuweisungsvorlage sollte wie folgt aussehen:

```

{"id":"123","count":"3","things":{"\"1\":{},\"2\":{},\"3\":{}}}

```

Variablen für Datentransformationen für API Gateway

Wenn Sie eine Parameterzuweisung erstellen, können Sie Kontextvariablen als Datenquelle verwenden. Wenn Sie Zuweisungsvorlagen-Transformationen erstellen, können Sie Kontextvariablen sowie input- und util-Variablen in Skripten verwenden, die Sie in der [Velocity Template Language \(VTL\)](#) schreiben. Beispiele für Zuweisungsvorlagen, die diese Referenzvariablen verwenden, finden Sie unter [the section called “Beispiele für die Verwendung von Variablen bei Zuweisungsvorlagen-Transformationen”](#).

Eine Liste der Referenzvariablen für Zugriffsprotokollierung finden Sie unter [the section called “Variablen für die Zugriffsprotokollierung in API Gateway”](#).

Kontextvariablen für Datentransformationen

Für Datenumwandlungen können Sie die folgenden `$context`-Variablen verwenden, bei denen die Groß-/Kleinschreibung beachtet werden muss.

Parameter	Description
<code>\$context.accountId</code>	Die Konto-ID des API-Besitzers. AWS
<code>\$context.apiId</code>	Die ID, die API Gateway Ihrer API zuweist.
<code>\$context.authorizer.claims. <i>property</i></code>	Eine Eigenschaft der Claims, die aus dem Amazon Cognito-Benutzerpool zurückgegeben werden, nachdem der Methodenaufruf erfolgreich authentifiziert wurde. Weitere Informationen finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”. Note Bei einem Aufruf von <code>\$context.authorizer.claims</code> wird null (0) zurückgegeben.
<code>\$context.authorizer.principalId</code>	Die ID des Prinzipalbenutzers in Verbindung mit dem Token, das vom Client gesendet und von einem API Gateway-Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bekannt) zurückgegeben wurde. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden” .
<code>\$context.authorizer. <i>property</i></code>	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der context-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der

Parameter	Description
	Genehmiger gibt folgende context-Zuweisung zurück: <pre data-bbox="829 331 1507 569"> "context" : { "key": "value", "numKey": 1, "boolKey": true } </pre> Ein Aufruf von <code>\$context.authorize r.key</code> gibt die Zeichenfolge "value" zurück, ein Aufruf von <code>\$context.authorize r.numKey</code> gibt die Zeichenfolge "1" zurück und ein Aufruf von <code>\$context.authorize r.boolKey</code> gibt die Zeichenfolge "true" zurück. Denn <i>property</i> das einzige unterstützte Sonderzeichen ist der (<code>_</code>) Unterstrich. Weitere Informationen finden Sie unter the section called “Lambda-Genemiger verwenden”.
<code>\$context.awsEndpointRequestId</code>	Die Anforderungs-ID des AWS Endpunkts.
<code>\$context.deploymentId</code>	Die ID der API-Bereitstellung
<code>\$context.domainName</code>	Der zum Aufrufen der API verwendete vollständige Domänennamen. Dieser Wert sollte mit dem für den eingehenden Host-Header übereinstimmen.
<code>\$context.domainPrefix</code>	Das erste Label der <code>\$context.domainName</code> .

Parameter	Description
<code>\$context.error.message</code>	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält. Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponse Body-Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und für die Zugriffsprotokollierung verwendet werden. Weitere Informationen erhalten Sie unter the section called “Kennzahlen” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen” .
<code>\$context.error.messageString</code>	Der Wert von <code>\$context.error.message</code> in Anführungszeichen, d. h. " <code>\$context.error.message</code> ".
<code>\$context.error.responseType</code>	Ein Typ von. GatewayResponse Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponse Body-Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und für die Zugriffsprotokollierung verwendet werden. Weitere Informationen erhalten Sie unter the section called “Kennzahlen” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen” .
<code>\$context.error.validationErrorString</code>	Eine Zeichenfolge mit einer detaillierten Validierungs-Fehlermeldung.
<code>\$context.extendedRequestId</code>	Die erweiterte ID, die API Gateway generiert und der API-Anfrage zuweist. Die erweiterte Anforderungs-ID enthält zusätzliche nützliche Informationen für Debugging und Fehlerbehebung.

Parameter	Description
<code>\$context.httpMethod</code>	Die verwendete HTTP-Methode. Gültige Werte sind: DELETE, GET, HEAD, OPTIONS, PATCH, POST und PUT.
<code>\$context.identity.accountId</code>	Die der AWS Anfrage zugeordnete Konto-ID.
<code>\$context.identity.apiKey</code>	Bei API-Methoden, für die ein API-Schlüssel erforderlich ist, ist diese Variable der API-Schlüssel für die Methodenanforderung. Bei Methoden, für die kein API-Schlüssel erforderlich ist, ist diese Variable nichtig. Weitere Informationen finden Sie unter the section called “Nutzungspläne” .
<code>\$context.identity.apiKeyId</code>	Die API-Schlüssel-ID für die API-Anforderung, falls ein API-Schlüssel erforderlich ist.
<code>\$context.identity.caller</code>	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Ressourcen unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Description
<code>\$context.identity.cognitoAuthenticationProvider</code>	Eine durch Komma getrennte Liste aller Amazon-Cognito-Authentifizierungsanbieter, die vom anfordernden Aufrufer verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, <code>cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Weitere Informationen zu verfügbaren Amazon-Cognito-Authentifizierungsanbietern finden Sie unter Verbundidentitäten verwenden im Amazon-Cognito-Entwicklerhandbuch.
<code>\$context.identity.cognitoAuthenticationType</code>	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind <code>authenticated</code> für authentifizierte Identitäten und <code>unauthenticated</code> für nicht authentifizierte Identitäten.
<code>\$context.identity.cognitoIdentityId</code>	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Description
<code>\$context.identity.cognitoId</code> <code>entityPoolId</code>	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.principalOrgId</code>	Die AWS -Organisations-ID .
<code>\$context.identity.sourceIp</code>	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an den API-Gateway-Endpunkt gesendet wird.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Das PEM-encoded Client-Zertifikat, das der Client bei der gegenseitigen TLS-Authentifizierung vorgelegt hat. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domain-Namens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
<code>\$context.identity.clientCertificate.subjectDN</code>	Der Distinguished Name des Zertifikatantragsellers, den ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domain-Namens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
<code>\$context.identity.clientCertificate.issuerDN</code>	Der Distinguished Name des Ausstellers des Zertifikats, das ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.

Parameter	Description
<code>\$context.identity.clientCertificate.serialNumber</code>	Die Seriennummer des Zertifikats. Vorhanden , wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Das Datum, vor dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden , wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Das Datum, nach dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden , wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
<code>\$context.identity.vpcId</code>	Die VPC-ID der VPC, deren Anforderung an den API-Gateway-Endpunkt gesendet wird.
<code>\$context.identity.vpcEndpointId</code>	Die VPC-Endpoint-ID des VPC-Endpunkts, dessen Anforderung an den API-Gateway-Endpunkt gesendet wird. Diese ist nur vorhanden, wenn Ihre API privat ist.
<code>\$context.identity.user</code>	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Ressourcen unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Description
<code>\$context.identity.userAgent</code>	Die User-Agent -Kopfzeile des API-Aufrufers.
<code>\$context.identity.userArn</code>	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.isCanaryRequest</code>	Gibt <code>true</code> zurück, wenn die Anforderung an den Canary gerichtet war oder <code>false</code> , wenn die Anforderung nicht an den Canary ging. Dies ist nur vorhanden, wenn Sie einen Canary aktiviert haben.
<code>\$context.path</code>	Der Anforderungspfad. Bei einer Nicht-Proxy-Anforderungs-URL von <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> lautet der <code>\$context.path</code> -Wert beispielsweise <code>{stage}/root/child</code> .
<code>\$context.protocol</code>	Das Anforderungsprotokoll ist z. B. HTTP/1.1.

 Note

API Gateway-APIs können HTTP/2 Anfragen annehmen, aber API Gateway sendet Anfragen an Backend-Integrationen über HTTP/1.1. Daher wird das Anforderungsprotokoll so protokolliert, HTTP/1.1 als ob ein Client eine Anfrage sendet, die HTTP/2

Parameter	Description
<code>\$context.requestId</code>	Eine ID für die Anforderung. Clients können diese Anforderungs-ID überschreiben. Verwenden von <code>\$context.extendedRequestId</code> für eine eindeutige Anforderungs-ID, die API Gateway generiert.
<code>\$context.requestOverride.header. <i>header_name</i></code>	Der Anforderungs-Header-Override. Wenn dieser Parameter definiert ist, enthält er die Header, die statt der HTTP Header, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestOverride.path. <i>path_name</i></code>	Der Anforderungspfad-Override. Wenn dieser Parameter definiert ist, enthält er den Anforderungspfad, der statt der URL-Pfadparameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestOverride.querystring. <i>querystring_name</i></code>	Der Abfragestring-Override. Wenn dieser Parameter definiert ist, enthält er die Abfragestrings, die statt der URL-Abfragestring-Parameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .

Parameter	Description
<code>\$context.responseOverride.header. <i>header_name</i></code>	Der Antwort-Header-Override. Wenn dieser Parameter definiert ist, enthält er den Header, der anstelle des Antwort-Headers, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.responseOverride.status</code>	Der Antwortstatuscode-Override. Wenn dieser Parameter definiert ist, enthält er den Statuscode, der anstelle des Methoden-Antwortstatus, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestTime</code>	Die Anforderungszeit im CLF -Format (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	Die Anforderungszeit im Epoch -Format in Millisekunden.
<code>\$context.resourceId</code>	Der Bezeichner, den API Gateway Ihrer Ressource zuweist.

Parameter	Description
<code>\$context.resourcePath</code>	Der Pfad zu Ihrer Ressource. Beim Nicht-Proxy-Anforderungs-URI von <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> lautet der <code>\$context.resourcePath</code> - Wert beispielsweise <code>/root/child</code> . Weitere Informationen finden Sie unter Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen .
<code>\$context.stage</code>	Die Bereitstellungsstufe der API-Anforderung (z. B. Beta oder Prod).
<code>\$context.wafResponseCode</code>	Die von AWS WAF empfangene Antwort: <code>WAF_ALLOW</code> oder <code>WAF_BLOCK</code> . Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called "AWS WAF" .
<code>\$context.webaclArn</code>	Vollständiger ARN der Web-Zugriffskontrollliste (Web-ACL), anhand deren entschieden wird, ob die Anforderung zugelassen oder blockiert wird. Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called "AWS WAF" .

Eingabevariablen

Sie können die folgenden `$input`-Variablen, bei denen die Groß- und Kleinschreibung beachtet werden muss, verwenden, um auf die Nutzdaten der Methodenanforderung und die Parameter der Methodenanforderung zu verweisen. Die folgenden Funktionen sind verfügbar:

Variable und Funktion	Beschreibung
<code>\$input.body</code>	Gibt die Nutzlast der Raw-Anforderung als Zeichenfolge zurück. Sie können <code>\$input.body</code> verwenden, um ganze Fließkommazahlen beizubehalten, z. B. <code>10.00</code>.
<code>\$input.json(x)</code>	Diese Funktion wertet einen JSONPath-Ausdruck aus und gibt die Ergebnisse als JSON-Zeichenfolge zurück. Beispielsweise gibt <code>\$input.json('\$.pets')</code> eine JSON-Zeichenfolge zurück, die die <code>pets</code>-Struktur abbildet. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.
<code>\$input.params()</code>	Gibt die Zuweisung aller Anforderungsparameter zurück. Wir empfehlen, das Ergebnis <code>\$util.escapeJavaScript</code> zu bereinigen, um einen möglichen Injektionsangriff zu vermeiden. Um die vollständige Kontrolle über die Bereinigung von Anfragen zu erhalten, verwenden Sie eine Proxy-Integration ohne Vorlage und übernehmen Sie die Anforderungsbereinigung in Ihrer Integration.
<code>\$input.params(x)</code>	Gibt aus der Zeichenfolge eines Parameternamens <code>x</code> aus dem Pfad, der Abfragezeichenfolge oder dem Header-Wert (in dieser Reihenfolge) den Wert eines Methodenanforderungs-Parameters zurück. Wir empfehlen, den Parameter <code>\$util.escapeJavaScript</code> zu bereinigen, um einen möglichen Injektionsangriff zu vermeiden. Um die vollständige Kontrolle über die Parameterbereinigung zu erhalten, verwenden Sie eine

Variable und Funktion	Beschreibung
	Proxy-Integration ohne Vorlage und übernehmen Sie die Anforderungsbereinigung in Ihrer Integration.
<code>\$input.path(x)</code>	Empfängt eine JSONPath-Ausdruckszeichenfolge (x) und gibt eine JSON-Objektdarstellung des Ergebnisses zurück. Dies ermöglicht einen nativen Zugriff auf Elemente der Nutzlast in Apache Velocity Template Language (VTL) und deren Bearbeitung. Beispiel: Der Ausdruck <code>\$input.path('\$\$.pets')</code> könnte das folgende Objekt zurückgeben: <pre>[{ "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 }]</pre> <code>\$input.path('\$\$.pets').size()</code> gibt "3" zurück. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.

Stufenvariablen

Sie können die folgenden Stufenvariablen als Platzhalter für ARNs und URLs in Methodenintegrationen verwenden. Weitere Informationen finden Sie unter [the section called “Verwenden von Stufenvariablen”](#).

Syntax	Description
<code>\$stageVariables. <i>variable_name</i> ,</code> <code>\$stageVariables[' <i>variable_name</i> ']</code> oder <code>\${stageVariables[' <i>variable_name</i> ']}</code>	<i>variable_name</i> steht für den Namen einer Stufenvariablen.

Util-Variablen

Sie können die folgenden `$util`-Variablen verwenden, bei denen Groß- und Kleinschreibung beachtet werden muss, um Dienstprogrammfunktionen für Zuweisungsvorlagen zu verwenden. Sofern nicht anders angegeben, ist der Standardzeichensatz UTF-8.

Funktion	Description
<code>\$util.escapeJavaScript()</code>	Escapiert die Zeichen in einer Zeichenfolge mithilfe von JavaScript Zeichenfolgenregeln.

Note

Mit dieser Funktion werden alle einfachen Anführungszeichen (') durch Escape-Zeichen (\ ') geschützt. Allerdings sind diese durch Escape-Zeichen geschützten einfachen Anführungszeichen in JSON nicht zulässig. Sofern die Ausgabe dieser Funktion in einer JSON-Eigenschaft verwendet werden soll, müssen alle einfachen Anführungszeichen, die durch Escape-Zeichen geschützt sind (\ '), wieder in

Funktion	Description
	reguläre einfache Anführungszeichen (') geändert werden. Das wird im folgenden Beispiel veranschaulicht: <pre data-bbox="911 380 1476 537">"input" : "\$util.escapeJavaScript(<i>data</i>).replaceAll("\\'", "'")"</pre>
<code>\$util.parseJson()</code>	Erhält das "stringify"-JSON-Objekt und gibt eine Objektdarstellung des Ergebnisses zurück. Mit dem Ergebnis dieser Funktion können Sie Elemente der Nutzlast, die in Apache Velocity Template Language (VTL) sind, aufrufen und bearbeiten. Angenommen, Sie haben folgende Nutzlast: <pre data-bbox="834 968 1507 1087">{ "errorMessage": "{ \"key1\": \"var1\", \"key2\": { \"arr\": [1, 2, 3] } } }</pre> Und verwenden die folgende Mapping-Vorlage: <pre data-bbox="834 1199 1507 1514">#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$.errorMessage'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> Dann erhalten Sie die folgende Ausgabe: <pre data-bbox="834 1619 1507 1780">{ "errorMessageObjKey2ArrVal" : 1 }</pre>

Funktion	Description
<code>\$util.urlEncode()</code>	Konvertiert eine Zeichenfolge in das Format "application/x-www-form-urlencoded".
<code>\$util.urlDecode()</code>	Dekodiert eine "application/x-www-form-urlencoded" - Zeichenfolge. application/x
<code>\$util.base64Encode()</code>	Codiert die Daten in eine base64-verschlüsselte Zeichenfolge.
<code>\$util.base64Decode()</code>	Decodiert die Daten einer base64-verschlüsselten Zeichenfolge.

Gateway-Antworten für REST APIs in API Gateway

Eine Gateway-Antwort wird durch einen Antworttyp identifiziert, der durch API Gateway definiert ist. Die Antwort besteht aus einem HTTP-Statuscode, einer Reihe zusätzlicher Header, die über Parameterzuweisungen festgelegt werden, und Nutzdaten, die mithilfe einer Nicht-[VTL](#)-Zuweisungsvorlage generiert werden.

In der API Gateway REST API wird eine Gateway-Antwort durch die dargestellt [GatewayResponse](#). In OpenAPI wird eine GatewayResponse Instanz durch die Erweiterung [x-amazon-apigateway-gateway-responses.GatewayResponse](#) beschrieben.

Zum Aktivieren einer Gateway-Antwort richten Sie sie für einen [unterstützten Antworttyp](#) auf API-Ebene ein. Wenn API Gateway eine Antwort dieses Typs zurückgibt, werden die in der Gateway-Antwort definierten Header-Mappings und Nutzlast-Mapping-Vorlagen angewendet, um die zugewiesenen Ergebnisse an den API-Aufrufer zurückzugeben.

Im folgenden Abschnitt wird gezeigt, wie Gateway-Antworten mit Hilfe der API Gateway-Konsole und der API Gateway-REST-API eingerichtet werden.

Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen

Wenn API Gateway eine eingehende Anfrage nicht verarbeiten kann, wird eine Fehlermeldung an den Client gesendet, ohne die Anfrage an das Integrations-Backend weiterzuleiten. Standardmäßig enthält diese Meldung eine kurze Beschreibung des Fehlers. Wenn Sie beispielsweise versuchen,

einen Vorgang für eine nicht definierte API-Ressource aufzurufen, erhalten Sie eine Fehlermeldung mit folgendem Inhalt: { "message": "Missing Authentication Token" }. Wenn Sie API Gateway noch nicht kennen, ist die Ursache eines Problems möglicherweise schwer zu ermitteln.

Bei einigen Fehlerantworten ermöglicht API Gateway die Anpassung durch API-Entwickler. So können Antworten in verschiedenen Formaten zurückgegeben werden. Bei dem Beispiel `Missing Authentication Token` können Sie der ursprünglichen Antwortnutzlast einen Hinweis auf die Fehlerursache hinzufügen, wie in diesem Beispiel: { "message": "Missing Authentication Token", "hint": "The HTTP method or resources may not be supported." }.

Wenn Ihre API zwischen einer externen Börse und der AWS Cloud vermittelt, verwenden Sie VTL-Mapping-Vorlagen für Integrationsanfragen oder Integrationsantworten, um die Nutzlast von einem Format einem anderen Format zuzuordnen. Die VTL-Mapping-Vorlagen funktionieren aber nur für gültige Anforderungen mit erfolgreichen Antworten.

Bei ungültigen Anfragen umgeht API Gateway die Integration vollständig und gibt eine Fehlerantwort zurück. Sie müssen die Anpassung für die Fehlermeldungen nutzen, um diese in einem austauschkonformen Format darzustellen. Hier wird die Anpassung in einer anderen als der VTL-Mapping-Vorlage dargestellt, die nur einfache Variablenersetzungen unterstützt.

Wir bezeichnen die von API Gateway generierte Fehlerantwort auf alle von API Gateway generierten Antworten allgemein als Gateway-Antworten. Dies unterscheidet die von API Gateway generierten Antworten von den Integrationsantworten. Eine Mapping-Vorlage für eine Gateway-Antwort kann auf `$context`-Variablenwerte und `$stageVariables`-Eigenschaftswerte sowie auf Methodenanforderungs-Parameter im Format `method.request.param-position.param-name` zugreifen.

Weitere Informationen zu `$context`-Variablen finden Sie unter [Kontextvariablen für Datentransformationen](#). Mehr über `$stageVariables` erfahren Sie unter [Stufenvariablen](#). Weitere Informationen zu Methodenanforderungs-Parametern finden Sie unter [the section called "Eingabevariablen"](#).

Themen

- [Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole](#)
- [Gateway-Antwort mit der API Gateway REST-API einrichten](#)
- [Einrichten der Gateway-Antwortanpassung in OpenAPI](#)
- [Gateway-Antworttypen für API Gateway](#)

Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole

Im folgenden Beispiel sehen Sie, wie Sie eine Gateway-Antwort für eine REST-API mithilfe der API-Gateway-Konsole einrichten

So passen Sie eine Gateway-Antwort über die API Gateway-Konsole an:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Gateway-Antworten.
4. Wählen Sie einen Antworttyp und dann Bearbeiten aus. In dieser Anleitung verwenden wir Fehlendes Authentifizierungstoken als Beispiel.
5. Sie können den von API Gateway generierten Statuscode ändern, um einen anderen Statuscode auszugeben, der den Anforderungen Ihrer API entspricht. In diesem Beispiel ändert die Anpassung den Statuscode vom Standard (403) zu 404, da diese Fehlermeldung auftritt, wenn ein Client eine nicht unterstützte oder ungültige Ressource aufruft, die als nicht gefunden betrachtet werden kann.
6. Um zu den benutzerdefinierten Headern zurückzukehren, wählen Sie Header hinzufügen hinzufügen unter Antwort-Header aus. Zur Veranschaulichung fügen wir die folgenden benutzerdefinierten Header hinzu:

```
Access-Control-Allow-Origin: 'a.b.c'
x-request-id:method.request.header.x-amzn-RequestId
x-request-path:method.request.path.petId
x-request-query:method.request.querystring.q
```

In den vorstehenden Header-Mappings wird dem 'a.b.c'-Header ein statischer Domänenname (Allow-Control-Allow-Origin) hinzugefügt, um den CORS-Zugriff auf die API zuzulassen; der x-amzn-RequestId in der Antwort wird der Eingabeanforderungs-Header request-id zugewiesen; dem petId-Header in der Antwort wird die request-path-Pfadvariable zugewiesen und dem q-Header der Antwort wird der request-query-Abfrageparameter der ursprünglichen Anforderung zugewiesen.

7. Behalten Sie unter Antwortvorlagen die Option application/json für Inhaltstyp bei und geben Sie die folgende Text-Mapping-Vorlage in den Editor Vorlagentext ein:

```
{
```

```
"message": "$context.error.messageString",
"type": "$context.error.responseType",
"statusCode": "'404'",
"stage": "$context.stage",
"resourcePath": "$context.resourcePath",
"stageVariables.a": "$stageVariables.a"
}
```

In diesem Beispiel sehen Sie, wie die `$context`- und `$stageVariables`-Eigenschaften den Eigenschaften des Gateway-Antworttexts zugewiesen werden.

8. Wählen Sie **Änderungen speichern** aus.
9. Stellen Sie die API in einer neuen oder vorhandenen Stufe bereit.

Testen Sie Ihre Gateway-Antwort, indem Sie den folgenden CURL-Befehl aufrufen.

Dabei wird davon ausgegangen, dass die Aufruf-URL der entsprechenden API-Methode

`https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/{petId}` ist:

```
curl -v -H 'x-amzn-RequestId:123344566' https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/5/type?q=1
```

Da der zusätzliche Abfragezeichenfolgeparameter `q=1` nicht mit der API kompatibel ist, wird ein Fehler in der angegebene Gateway-Antwort zurückgegeben. Sie sollten eine Gateway-Antwort wie die folgende erhalten:

```
> GET /custErr/pets/5?q=1 HTTP/1.1
Host: o81lxisefl.execute-api.us-east-1.amazonaws.com
User-Agent: curl/7.51.0
Accept: */*

HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: 334
Connection: keep-alive
Date: Tue, 02 May 2017 03:15:47 GMT
x-amzn-RequestId: 123344566
Access-Control-Allow-Origin: a.b.c
x-amzn-ErrorType: MissingAuthenticationTokenException
header-1: static
x-request-query: 1
```



```
x-request-path: 5
X-Cache: Error from cloudfront
Via: 1.1 441811a054e8d055b893175754efd0c3.cloudfront.net (CloudFront)
X-Amz-Cf-Id: nNDR-fX4csbRoAgtQJ16u0rTDz9FZWT-Mk93KgoxnfzDlTUh3flmzA==

{
  "message": "Missing Authentication Token",
  "type": MISSING_AUTHENTICATION_TOKEN,
  "statusCode": '404',
  "stage": custErr,
  "resourcePath": /pets/{petId},
  "stageVariables.a": a
}
```

Im vorherigen Beispiel wird davon ausgegangen, dass das API-Backend [Pet Store](#) ist und die API über die definierte Stufenvariable `a` verfügt.

Gateway-Antwort mit der API Gateway REST-API einrichten

Bevor Sie eine Gateway-Antwort mit der API Gateway-REST-API anpassen können, müssen Sie eine API erstellt und deren ID abgerufen haben. Zum Abrufen der API-ID können Sie der [restapi:gateway-responses](#)-Link-Beziehung folgen und das Ergebnis untersuchen.

So passen Sie eine Gateway-Antwort mit der API Gateway-REST-API an:

1. Um eine gesamte [GatewayResponse](#)-Instanz zu überschreiben, rufen Sie die Aktion [gatewayresponse:put](#) auf. Geben Sie im URL-Pfadparameter einen gewünschten [responseType](#) an und stellen Sie in der Anforderungsnutzlast die Mappings [statusCode](#), [responseParameters](#) und [responseTemplates](#) bereit.
2. Um einen Teil einer GatewayResponse-Instance zu aktualisieren, rufen Sie die Aktion [gatewayresponse:update](#) auf. Geben Sie einen gewünschten `responseType` im URL-Pfadparameter an und stellen Sie in der Anfragenutzlast die einzelnen gewünschten GatewayResponse-Eigenschaften bereit, z. B. das Mapping `responseParameters` oder `responseTemplates`.

Einrichten der Gateway-Antwortanpassung in OpenAPI

Mit der `x-amazon-apigateway-gateway-responses`-Erweiterung auf der API-Stammebene können Sie Gateway-Antworten in OpenAPI anpassen. Die folgende OpenAPI-Definition zeigt ein Beispiel für die Anpassung [GatewayResponse](#) des `MISSING_AUTHENTICATION_TOKEN` Typs.

```

"x-amazon-apigateway-gateway-responses": {
  "MISSING_AUTHENTICATION_TOKEN": {
    "statusCode": 404,
    "responseParameters": {
      "gatewayresponse.header.x-request-path": "method.input.params.petId",
      "gatewayresponse.header.x-request-query": "method.input.params.q",
      "gatewayresponse.header.Access-Control-Allow-Origin": "'a.b.c'",
      "gatewayresponse.header.x-request-header": "method.input.params.Accept"
    },
    "responseTemplates": {
      "application/json": "{\n    \"message\": $context.error.messageString,\n    \"type\": \"$context.error.responseType\",\n    \"stage\": \"$context.stage\",\n    \"resourcePath\": \"$context.resourcePath\",\n    \"stageVariables.a\": \"$stageVariables.a\",\n    \"statusCode\": \"'404'\"\n  }"
    }
  }
}

```

In diesem Beispiel wird durch die Anpassung der Standardstatuscode 403 in 404 geändert. Außerdem werden der Gateway-Antwort vier Header-Parameter und eine Text-Mapping-Vorlage für den `application/json`-Medientyp hinzugefügt.

Gateway-Antworttypen für API Gateway

API Gateway stellt die folgenden Gateway-Antworten zur Anpassung durch API-Entwickler zur Verfügung.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
ACCESS_DENIED	403	Die Gateway-Antwort bei Autorisierungsfehlern, beispielsweise bei einer Verweigerung des Zugriffs durch einen benutzerdefinierten oder Amazon Cognito-Genehmiger. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
API_CONFIGURATION_ERROR	500	Die Gateway-Antwort für eine ungültige API-Konfiguration, einschließlich wenn eine ungültige Endpunktadresse übermittelt wird, wenn die base64-Decodierung von Binärdaten bei Verwendung der binären Unterstützung fehlschlägt oder wenn das Integrationsantwort-Mapping keiner Vorlage entspricht und keine Standardvorlage konfiguriert ist. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_5XX für diese Antwort verwendet.
AUTHORIZER_CONFIGURATION_ERROR	500	Die Gateway-Antwort für eine fehlgeschlagene Verbindung zu einem benutzerdefinierten oder Amazon Cognito-Genehmiger. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_5XX für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
AUTHORIZER_FAILURE	500	Die Gateway-Antwort, wenn ein benutzerdefinierter oder Amazon Cognito-Genehmiger den Aufrufer nicht authentifizieren konnte. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_5XX</code> für diese Antwort verwendet.
BAD_REQUEST_PARAMETERS	400	Die Gateway-Antwort, wenn der Anforderungsparameter nicht gemäß einem aktivierten Anforderungsvalidierer überprüft werden kann. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
BAD_REQUEST_BODY	400	Die Gateway-Antwort, wenn der Anforderungstext nicht gemäß einem aktivierten Anforderungsvalidierer überprüft werden kann. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
DEFAULT_4XX	Null	Die Standard-Gateway-Antwort für einen nicht angegebenen Antworttyp mit dem Statuscode 4XX. Durch eine Änderung des Statuscodes dieser Fallback-Gateway-Antwort werden die Statuscodes von allen anderen 4XX-Antworten in den neuen Wert geändert. Mit einem Zurücksetzen des Statuscodes auf null werden die Statuscodes von sämtlichen anderen 4XX-Antworten auf ihre ursprünglichen Werte zurückgesetzt.  Note AWS WAF benutzerdefinierte Antworten haben Vorrang vor benutzerdefinierten Gateway-Antworten.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
DEFAULT_5XX	Null	Die Standard-Gateway-Antwort für einen nicht angegebenen Antworttyp mit dem Statuscode 5XX. Durch eine Änderung des Statuscodes dieser Fallback-Gateway-Antwort werden die Statuscodes von allen anderen 5XX-Antworten in den neuen Wert geändert. Mit einem Zurücksetzen des Statuscodes auf null werden die Statuscodes von sämtlichen anderen 5XX-Antworten auf ihre ursprünglichen Werte zurückgesetzt.
EXPIRED_TOKEN	403	Fehler bei der Gateway-Antwort auf ein AWS Authentifizierungstoken ist abgelaufen. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
INTEGRATION_FAILURE	504	Die Gateway-Antwort bei einem Integrationsfehler. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_5XX für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
INTEGRATION_TIMEOUT	504	Die Gateway-Antwort, wenn das Zeitlimit für die Integration überschritten wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_5XX</code> für diese Antwort verwendet.
INVALID_API_KEY	403	Die Gateway-Antwort, wenn für eine Methode, die einen API-Schlüssel erfordert, ein ungültiger API-Schlüssel übermittelt wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
INVALID_SIGNATURE	403	Die Gateway-Antwort auf einen ungültigen AWS Signaturfehler. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
MISSING_AUTHENTICATION_TOKEN	403	Die Gateway-Antwort, wenn das Authentifizierungstoken fehlt. Dazu zählen auch die Fälle, bei denen der Client versucht, eine nicht unterstützte API-Methode oder -Ressource aufzurufen. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
QUOTA_EXCEEDED	429	Die Gateway-Antwort, wenn das Kontingent für den Nutzungsplan überschritten wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
REQUEST_TOO_LARGE	413	Die Gateway-Antwort, wenn die Anforderung zu groß ist. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>HTTP content length exceeded 10485760 bytes</code> für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
RESOURCE_NOT_FOUND	404	Die Gateway-Antwort, wenn Amazon API Gateway die angegebene Ressource nicht finden kann, nachdem eine API-Anfrage die Authentifizierung und Autorisierung – mit Ausnahme der API-Schlüssel-Authentifizierung und -Autorisierung – durchlaufen hat. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
THROTTLED	429	Die Gateway-Antwort, wenn Drosselungslimits auf Nutzungsplan-, Methoden-, Stufen- oder Kontoebene überschritten wurden. Wenn kein Antworttyp angegeben ist, wird standardmäßig <code>DEFAULT_4XX</code> für diese Antwort verwendet.
UNAUTHORIZED	401	Die Gateway-Antwort, wenn der benutzerdefinierte oder Amazon Cognito-Genehmiger den Aufrufer nicht authentifizieren konnte.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
UNSUPPORTED_MEDIA_TYPE	415	Die Gateway-Antwort, wenn strenges Passthrough-Verhalten aktiviert ist und eine Nutzlast einen nicht unterstützten Medientyp aufweist. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
WAF_FILTERED	403	Die Gateway-Antwort, wenn eine Anforderung von AWS WAF blockiert wird. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.

 **Note**

[AWS WAF benutzerdefinierte Antworten](#) haben Vorrang vor benutzerdefinierten Gateway-Antworten.

CORS für REST APIs im API Gateway

[Cross-Origin Resource Sharing \(CORS\)](#) ist eine Browser-Sicherheitsfunktion, die Cross-Origin-HTTP-Anfragen einschränkt, die von Skripts eingeleitet werden, die im Browser ausgeführt werden. Weitere Informationen finden Sie unter [Was ist CORS?](#).

Feststellung, ob die CORS-Unterstützung aktiviert werden soll

Cross-Origin-HTTP-Anfragen zielen auf Folgendes:

- Eine andere Domäne (z. B. von `example.com` an `amazondomains.com`)
- Eine andere Subdomäne (z. B. von `example.com` an `petstore.example.com`)
- Ein anderer Port (z. B. von `example.com` an `example.com:10777`)
- Ein anderes Protokoll (z. B. von `https://example.com` an `http://example.com`)

Wenn Sie nicht auf Ihre API zugreifen können und eine Fehlermeldung erhalten, die `Cross-Origin Request Blocked` enthält, müssen Sie möglicherweise CORS aktivieren.

Cross-Origin-HTTP-Anfragen lassen sich in zwei Arten unterteilen: einfache und nicht einfache Anfragen.

Aktivieren von CORS für eine einfache Anfrage

Eine HTTP-Anfrage ist einfach, wenn alle der folgenden Bedingungen erfüllt sind:

- Sie wird für eine API-Ressource gestellt, die nur GET-, HEAD- und POST-Anfragen erlaubt.
- Wenn es sich um eine POST-Methodenanfrage handelt, muss sie einen `Origin-Header` enthalten.
- Der Anfragenutzlast-Inhaltstyp ist `text/plain`, `multipart/form-data` oder `application/x-www-form-urlencoded`.
- Die Anfrage enthält keine benutzerdefinierten Header.
- Alle weiteren Anforderungen, die in der [Mozilla CORS-Dokumentation für einfache Anfragen](#) aufgelistet sind.

Bei einfachen quellenübergreifenden POST-Methodenanfragen muss die Antwort Ihrer Ressource den Header `Access-Control-Allow-Origin: '*'` oder `Access-Control-Allow-Origin: 'origin'` enthalten.

Alle anderen Cross-Origin-HTTP-Anfragen sind nicht einfache Anfragen.

Aktivieren von CORS für eine nicht einfache Anfrage

Wenn die Ressourcen Ihrer API nicht einfache Anfragen erhalten, müssen Sie je nach Integrationstyp zusätzliche CORS-Unterstützung aktivieren.

Aktivieren von CORS für Integrationen ohne Proxy

Für diese Integrationen erfordert das [CORS-Protokoll](#), dass der Browser eine Preflight-Anfrage an den Server sendet und auf die Genehmigung (oder Anforderung der Anmeldeinformationen) vom

Server wartet, bevor die tatsächliche Anfrage gesendet wird. Sie müssen Ihre API so konfigurieren, dass sie eine entsprechende Antwort auf die Preflight-Anfrage sendet.

So erstellen Sie eine Preflight-Antwort:

1. Erstellen Sie eine OPTIONS-Methode mit einer Mock-Integration.
2. Fügen Sie der Antwort der Methode 200 die folgenden Antwort-Header hinzu:
 - Access-Control-Allow-Headers
 - Access-Control-Allow-Methods
 - Access-Control-Allow-Origin
3. Setzen Sie das Integrations-Pass-Through-Verhalten auf NEVER. In diesem Fall wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit der Antwort „HTTP 415 Unsupported Media Type“ zurückgewiesen. Weitere Informationen finden Sie unter [Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway](#).
4. Geben Sie Werte für die Antwort-Header ein. Verwenden Sie die folgenden Header-Werte, um alle Ursprünge, alle Methoden und allgemeine Header zuzulassen:
 - Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'
 - Access-Control-Allow-Methods: 'DELETE,GET,HEAD,OPTIONS,PUT,POST,PATCH'
 - Access-Control-Allow-Origin: '*'

Nachdem Sie die Preflight-Anfrage erstellt haben, müssen Sie den Access-Control-Allow-Origin: '*' - oder Access-Control-Allow-Origin: '*origin*' -OR-Header für alle CORS-fähigen Methoden für mindestens alle 200 Antworten zurückgeben.

Aktivierung von CORS für Nicht-Proxy-Integrationen mit dem AWS-Managementkonsole

Sie können den verwenden, um CORS AWS-Managementkonsole zu aktivieren. API Gateway erstellt eine OPTIONS-Methode und fügt den Access-Control-Allow-Origin-Header zu Ihren vorhandenen Methodenintegrationsantworten hinzu. Dies funktioniert nicht immer und manchmal müssen Sie die Integrationsantwort manuell ändern, um den Access-Control-Allow-Origin-Header für alle CORS-fähigen Methoden für mindestens alle 200 Antworten zurückzugeben.

Wenn Sie für Ihre API binäre Medientypen auf */* festgelegt haben, ändern Sie beim Erstellen einer OPTIONS-Methode durch API Gateway `contentHandling` auf `CONVERT_TO_TEXT`.

Der folgende [update-integration](#)-Befehl ändert contentHandling für eine Integrationsanforderung auf CONVERT_TO_TEXT:

```
aws apigateway update-integration \  
  --rest-api-id abc123 \  
  --resource-id aaa111 \  
  --http-method OPTIONS \  
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

Mit dem folgenden [update-integration-response](#) Befehl wird das contentHandling in CONVERT_TO_TEXT einer Integrationsantwort geändert:

```
aws apigateway update-integration-response \  
  --rest-api-id abc123 \  
  --resource-id aaa111 \  
  --http-method OPTIONS \  
  --status-code 200 \  
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

Aktivieren der CORS-Unterstützung für Proxy-Integrationen

Bei einer Lambda-Proxy-Integration oder einer HTTP-Proxy-Integration ist Ihr Backend für die Rückgabe der Header Access-Control-Allow-Origin, Access-Control-Allow-Methods und Access-Control-Allow-Headers verantwortlich, da eine Proxy-Integration keine Integrationsantwort zurückgibt.

Die folgenden Beispiel-Lambda-Funktionen geben die erforderlichen CORS-Header zurück:

Node.js

```
export const handler = async (event) => {  
  const response = {  
    statusCode: 200,  
    headers: {  
      "Access-Control-Allow-Headers" : "Content-Type",  
      "Access-Control-Allow-Origin": "https://www.example.com",  
      "Access-Control-Allow-Methods": "OPTIONS,POST,GET"  
    },  
    body: JSON.stringify('Hello from Lambda!'),  
  };  
  return response;  
};
```

```
};
```

Python 3

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://www.example.com',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET'
        },
        'body': json.dumps('Hello from Lambda!')
    }
```

Themen

- [CORS für eine Ressource über die API Gateway-Konsole aktivieren](#)
- [CORS für eine Ressource über die API Gateway-Import-API aktivieren](#)
- [CORS für eine API-Gateway-API testen](#)

CORS für eine Ressource über die API Gateway-Konsole aktivieren

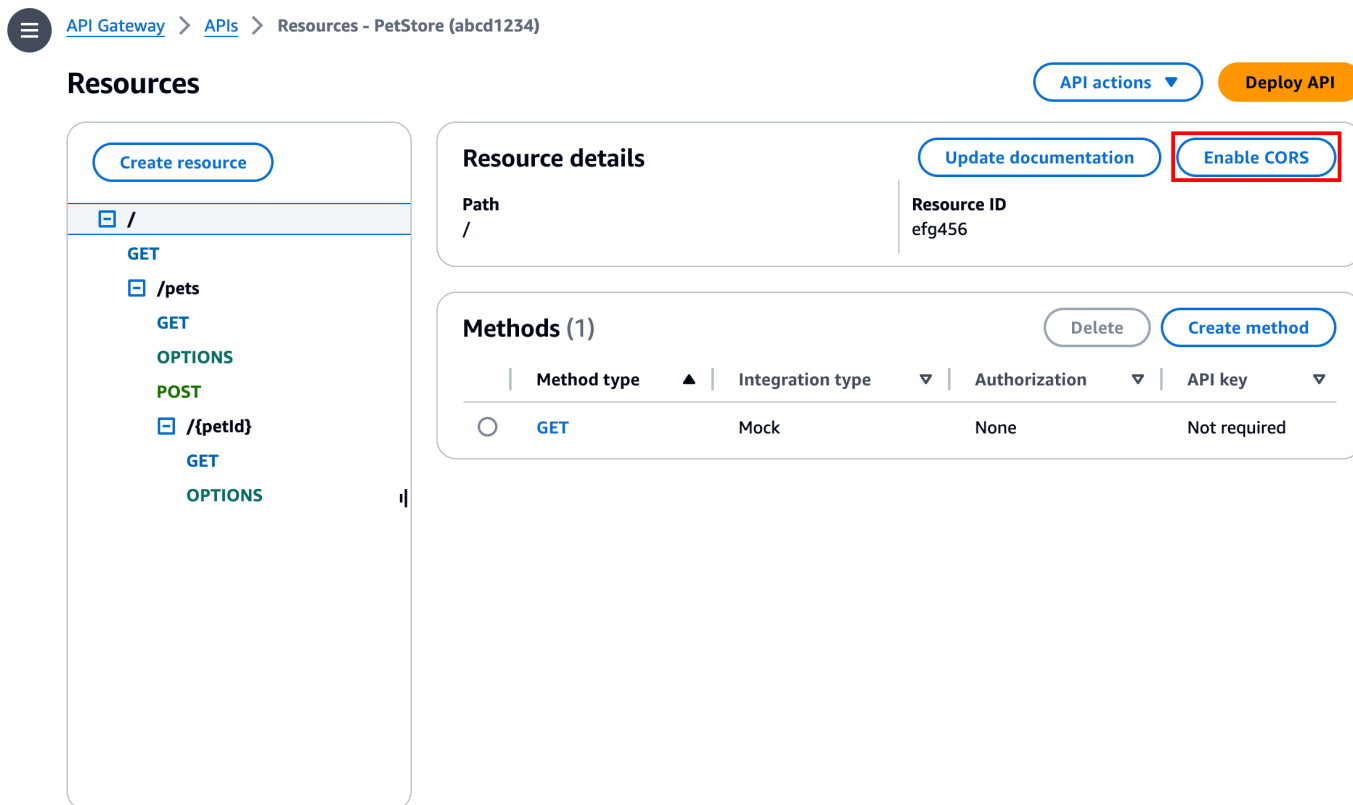
Sie können die API Gateway-Konsole verwenden, um die CORS-Unterstützung für eine oder alle Methoden einer von Ihnen erstellten REST-API-Ressource zu aktivieren. Nachdem Sie COR-Support aktiviert haben, setzen Sie das Pass-Through-Verhalten für die Integration auf NEVER. In diesem Fall wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit der Antwort „HTTP 415 Unsupported Media Type“ zurückgewiesen. Weitere Informationen finden Sie unter [Verhalten der Methodenanforderung für Nutzdaten ohne Zuweisungsvorlagen für REST-APIs in API Gateway](#).

Important

Ressourcen können untergeordnete Ressourcen enthalten. Die Aktivierung der CORS-Unterstützung für eine Ressource und ihre Methoden führt nicht dazu, dass sie rekursiv für untergeordnete Ressourcen und ihre Methoden aktiviert wird.

So aktivieren Sie die CORS-Unterstützung für eine REST-API-Ressource:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API aus.
3. Wählen Sie eine Ressource unter Resources.
4. Wählen Sie im Abschnitt Ressourcendetails die Option CORS aktivieren aus.



5. Gehen Sie im Feld CORS aktivieren wie folgt vor:
 - a. (Optional) Wenn Sie eine benutzerdefinierte Gateway-Antwort erstellt haben und die CORS-Unterstützung für eine Antwort aktivieren möchten, wählen Sie eine Gateway-Antwort aus.
 - b. Wählen Sie die einzelnen Methoden aus, um die CORS-Unterstützung zu aktivieren. Für die OPTION-Methode muss CORS aktiviert sein.

Wenn Sie die CORS-Unterstützung für eine ANY-Methode aktivieren, ist CORS für alle Methoden aktiviert.

- c. Geben Sie im Eingabefeld Access-Control-Allow-Headers eine statische Zeichenfolge einer durch Kommata getrennten Liste der Header ein, die der Client in der tatsächlichen Anforderung der Ressource senden muss. Verwenden Sie die von der Konsole bereitgestellte Header-Liste mit 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token' oder geben Sie Ihren eigenen Header an.
- d. Verwenden Sie den von der Konsole bereitgestellten Wert '*' als Access-Control-Allow-Origin-Header-Wert, um Zugriffsanforderungen von allen Quellen zuzulassen, oder geben Sie eine bestimmte Quelle an, die auf die Ressource zugreifen darf.
- e. Wählen Sie Speichern.

 [API Gateway](#) > [APIs](#) > [Resources - PetStore \(abcd1234\)](#) > Enable CORS

Enable CORS

CORS settings [Info](#)

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API. When you save your configuration, API Gateway replaces any existing CORS settings with your new configuration.

Gateway responses

API Gateway will configure CORS for the selected gateway responses.

☐ Default 4XX

☐ Default 5XX

Access-Control-Allow-Methods

☐ GET

☒ OPTIONS

Access-Control-Allow-Headers

API Gateway will configure CORS for the selected gateway responses.

Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token

Access-Control-Allow-Origin

Enter an origin that can access the resource. Use a wildcard '*' to allow any origin to access the resource.

*

► Additional settings

[Cancel](#)

[Save](#)

Important

Wenn Sie die oben genannten Anweisungen auf die ANY-Methode in einer Proxy-Integration anwenden, werden keine geltenden CORS-Header festgelegt. Stattdessen

muss Ihr Backend die relevanten CORS-Header zurückgeben, z. B. `Access-Control-Allow-Origin`.

Wenn CORS für die GET-Methode aktiviert wurde, wird eine OPTIONS-Methode zur Ressource hinzugefügt, wenn sie dort noch nicht vorhanden ist. Die 200-Antwort der OPTIONS-Methode wird automatisch so konfiguriert, dass sie die drei `Access-Control-Allow-*`-Header zurückgibt, um die Preflight-Handshakes zu erfüllen. Darüber hinaus ist die tatsächliche (GET-) Methode standardmäßig auch so konfiguriert, dass sie den `Access-Control-Allow-Origin`-Header auch in seiner 200-Antwort zurückgibt. Für andere Arten von Antworten müssen Sie sie manuell konfigurieren, sodass sie den `Access-Control-Allow-Origin`-Header mit "*" oder bestimmten Ursprüngen zurückgeben, wenn Sie den `Cross-origin access`-Fehler nicht zurückgeben möchten.

Wenn Sie die CORS-Unterstützung für Ihre Ressource aktiviert haben, müssen Sie die API für die neuen Einstellungen (erneut) bereitstellen, damit diese wirksam werden. Weitere Informationen finden Sie unter [the section called "Eine Bereitstellung auswählen"](#).

Note

Wenn Sie CORS-Support nach Ausführung dieser Schritte nicht für Ihre Ressource aktivieren können, empfehlen wir, Ihre CORS-Konfiguration mit der Beispiel-API-/pets-Ressource zu vergleichen. Informationen zum Erstellen der Beispiel-API finden Sie unter [the section called "Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels"](#).

CORS für eine Ressource über die API Gateway-Import-API aktivieren

Wenn Sie die [API Gateway-Import-API](#) verwenden, können Sie CORS-Unterstützung mit einer OpenAPI-Datei einrichten. Sie müssen zunächst eine OPTIONS-Methode in Ihrer Ressource definieren, die die erforderlichen Header zurückgibt.

Note

Webbrowser erwarten `Access-Control-Allow-Headers`, dass in jeder API-Methode, die CORS-Anfragen akzeptiert, `Access-Control-Allow-Origin` Header eingerichtet werden. Darüber

hinaus stellen einige Browser zuerst eine HTTP-Anforderung an eine OPTIONS-Methode in derselben Ressource und erwarten dann, dieselben Header zu erhalten.

Options-Beispielmethode

Das folgende Beispiel erstellt eine OPTIONS-Methode für eine Mock-Integration.

OpenAPI 3.0

```
/users:
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    tags:
      - CORS
    responses:
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Origin:
            schema:
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates:
        application/json: "{$\"statusCode\": 200}"
      passthroughBehavior: "never"
      responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
```

```
method.response.header.Access-Control-Allow-Origin: "'*'"
```

OpenAPI 2.0

```
/users:
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    consumes:
      - "application/json"
    produces:
      - "application/json"
    tags:
      - CORS
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates: "{\\"statusCode\\": 200}"
      passthroughBehavior: "never"
      responses:
        "default":
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers : "'Content-Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods : "'*'"
            method.response.header.Access-Control-Allow-Origin : "'*'"
    responses:
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Headers:
            type: "string"
          Access-Control-Allow-Methods:
            type: "string"
          Access-Control-Allow-Origin:
            type: "string"
```

Nachdem Sie die OPTIONS-Methode für Ihre Ressource konfiguriert haben, können Sie die erforderlichen Header zu den anderen Methoden in derselben Ressource, die CORS-Anforderungen akzeptieren muss, hinzufügen.

1. Deklarieren Sie Access-Control-Allow-Origin und Headers für die Antworttypen.

OpenAPI 3.0

```
responses:
  200:
    description: Default response for CORS method
    headers:
      Access-Control-Allow-Origin:
        schema:
          type: "string"
      Access-Control-Allow-Methods:
        schema:
          type: "string"
      Access-Control-Allow-Headers:
        schema:
          type: "string"
    content: {}
```

OpenAPI 2.0

```
responses:
  200:
    description: Default response for CORS method
    headers:
      Access-Control-Allow-Headers:
        type: "string"
      Access-Control-Allow-Methods:
        type: "string"
      Access-Control-Allow-Origin:
        type: "string"
```

2. Richten Sie im x-amazon-apigateway-integration-Tag das Mapping für diese Header auf Ihre statischen Werte ein:

OpenAPI 3.0

```
responses:
  default:
    statusCode: "200"
    responseParameters:
```

```

        method.response.header.Access-Control-Allow-Headers: "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
        method.response.header.Access-Control-Allow-Methods: "'*'"
        method.response.header.Access-Control-Allow-Origin: "'*'"
    responseTemplates:
        application/json: |
            {}

```

OpenAPI 2.0

```

responses:
  "default":
    statusCode: "200"
    responseParameters:
      method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
      method.response.header.Access-Control-Allow-Methods : "'*'"
      method.response.header.Access-Control-Allow-Origin : "'*'"

```

Beispiel-API

Im folgenden Beispiel wird eine vollständige API mit einer OPTIONS-Methode und einer GET-Methode mit einer HTTP-Integration erstellt.

OpenAPI 3.0

```

openapi: "3.0.1"
info:
  title: "cors-api"
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "/test"
paths:
  /:
    get:
      operationId: "GetPet"
      responses:
        "200":

```

```

        description: "200 response"
        headers:
            Access-Control-Allow-Origin:
                schema:
                    type: "string"
        content: {}
    x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
            default:
                statusCode: "200"
                responseParameters:
                    method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
        responses:
            "200":
                description: "200 response"
                headers:
                    Access-Control-Allow-Origin:
                        schema:
                            type: "string"
                    Access-Control-Allow-Methods:
                        schema:
                            type: "string"
                    Access-Control-Allow-Headers:
                        schema:
                            type: "string"
                content:
                    application/json:
                        schema:
                            $ref: "#/components/schemas/Empty"
    x-amazon-apigateway-integration:
        responses:
            default:
                statusCode: "200"
                responseParameters:
                    method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
                    method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-Amz-Date,Authorization,X-Api-Key'"
                    method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:

```

```

        application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
components:
  schemas:
    Empty:
      type: "object"

```

OpenAPI 2.0

```

swagger: "2.0"
info:
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
  title: "cors-api"
basePath: "/test"
schemes:
- "https"
paths:
  /:
    get:
      operationId: "GetPet"
      produces:
        - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
            passthroughBehavior: "never"
            type: "http"
      options:
        consumes:
          - "application/json"

```

```

produces:
- "application/json"
responses:
  "200":
    description: "200 response"
    schema:
      $ref: "#/definitions/Empty"
    headers:
      Access-Control-Allow-Origin:
        type: "string"
      Access-Control-Allow-Methods:
        type: "string"
      Access-Control-Allow-Headers:
        type: "string"
  x-amazon-apigateway-integration:
    responses:
      default:
        statusCode: "200"
        responseParameters:
          method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
          method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-Amz-Date,Authorization,X-Api-Key'"
          method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
definitions:
  Empty:
    type: "object"

```

CORS für eine API-Gateway-API testen

Sie können die CORS-Konfiguration Ihrer API testen, indem Sie Ihre API aufrufen und die CORS-Header in der Antwort überprüfen. Der folgende `curl`-Befehl sendet eine OPTIONS-Anforderung an eine bereitgestellte API.

```
curl -v -X OPTIONS https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}
```

```
< HTTP/1.1 200 OK
< Date: Tue, 19 May 2020 00:55:22 GMT
```



```
< Content-Type: application/json
< Content-Length: 0
< Connection: keep-alive
< x-amzn-RequestId: a1b2c3d4-5678-90ab-cdef-abc123
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Content-Type,Authorization,X-Amz-Date,X-API-Key,X-Amz-Security-Token
< x-amz-apigw-id: Abcd=
< Access-Control-Allow-Methods: DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT
```

Die Header `Access-Control-Allow-Origin`, `Access-Control-Allow-Headers` und `Access-Control-Allow-Methods` in der Antwort zeigen, dass die API CORS unterstützt. Weitere Informationen finden Sie unter [CORS für REST APIs im API Gateway](#).

Binäre Medientypen für REST APIs in API Gateway

In API Gateway haben die API-Anfrage und -Antwort einen Text-Payload oder einen Binär-Payload. Eine Text-Nutzlast ist eine UTF-8-codierte JSON-Zeichenfolge. Eine binäre Nutzlast ist jede beliebige andere außer einer Text-Nutzlast. Die binäre Nutzlast kann beispielsweise eine JPEG-Datei, eine GZip Datei oder eine XML-Datei sein. Welche API-Konfiguration zur Unterstützung von binären Medien erforderlich ist, hängt davon ab, ob Ihre API Proxy- oder Nicht-Proxyintegrationen verwendet.

Wenn Sie eine Proxy-Integration mit Payload-Response-Streaming verwenden, müssen Sie Ihre binären Medientypen nicht konfigurieren. Weitere Informationen finden Sie unter [the section called “Streamen der Integrationsantwort für Ihre Proxy-Integrationen”](#).

AWS Lambda Proxy-Integrationen

Um binäre Nutzlasten für AWS Lambda Proxy-Integrationen zu verarbeiten, müssen Sie die Antwort Ihrer Funktion base64-kodieren. Sie müssen das auch für Ihre API konfigurieren. [binaryMediaTypes](#) Die `binaryMediaTypes`-Konfiguration Ihrer API ist eine Liste von Inhaltstypen, die Ihre API als Binärdaten behandelt. Beispiele für binäre Medientypen umfassen `image/png` oder `application/octet-stream`. Sie können das Platzhalterzeichen (*) verwenden, um mehrere Medientypen abzudecken.

API Gateway verwendet den ersten `Accept`-Header von Clients, um zu bestimmen, ob eine Antwort binäre Medien zurückgeben soll. Um binäre Medien zurückzugeben, wenn Sie die Reihenfolge der `Accept`-Header-Werte nicht kontrollieren können, z. B. Anforderungen von einem Browser, setzen Sie die binären Medientypen der API auf `*/*`.

Einen Beispielcode finden Sie unter [the section called “Rückgabe binärer Medien aus einer Lambda-Proxy-Integration in API Gateway”](#).

Wenn Sie eine Lambda-Proxyintegration mit Payload Response-Streaming verwenden, müssen Sie Ihre binären Medientypen nicht konfigurieren. Weitere Informationen finden Sie unter [the section called “Einrichten einer Lambda-Proxyintegration mit Nutzdaten-Antwort-Streaming”](#).

Nicht-Proxy-Integrationen

Um binäre Payloads für Nicht-Proxy-Integrationen zu verarbeiten, fügen Sie die Medientypen der Liste der Ressource hinzu. [binaryMediaTypes](#) RestApi Die `binaryMediaTypes`-Konfiguration Ihrer API ist eine Liste von Inhaltstypen, die Ihre API als Binärdaten behandelt. Alternativ können Sie die [ContentHandling-Eigenschaften](#) für die [Integration und die IntegrationResponseRessourcen](#) festlegen. Der Wert `contentHandling` kann `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` oder undefiniert sein.

Note

Für MOCK oder private Integrationen wird das Festlegen der `contentHandling`-Eigenschaften in der AWS-Managementkonsole nicht unterstützt. Sie müssen das AWS CLI, oder ein SDK verwenden CloudFormation, um die `contentHandling` Eigenschaften festzulegen.

Je nach `contentHandling`-Wert sowie abhängig von der Tatsache, ob der `Content-Type`-Header der Antwort oder der `Accept`-Header der eingehenden Anfrage mit einem Eintrag in der Liste `binaryMediaTypes` übereinstimmt, kann Amazon API Gateway die unformatierten binären Bytes als base64-kodierte Zeichenfolge kodieren, eine base64-kodierte Zeichenfolge zurück in die Raw-Bytes dekodieren oder den Textkörper ohne Änderung weitergeben.

Um binäre Payloads für Ihre API in API Gateway zu unterstützen, müssen Sie die API wie folgt konfigurieren:

- Fügen Sie die gewünschten binären Medientypen zur `binaryMediaTypes` Liste auf der [RestApi](#) Ressource hinzu. Wenn diese Eigenschaft und die `contentHandling`-Eigenschaft nicht definiert sind, werden die Nutzlasten als UTF-8-kodierte JSON-Zeichenfolgen verarbeitet.
- Legen Sie die Eigenschaft `contentHandling` der Ressource [Integration](#) fest.
 - Damit die Anforderungsnutzlast von einer Base64-kodierten Zeichenfolge in ihr binäres Blob konvertiert wird, setzen Sie die Eigenschaft auf `CONVERT_TO_BINARY`.

- Damit die Anforderungsnutzlast von einem binären Blob in eine Base64-codierte Zeichenfolge konvertiert wird, setzen Sie die Eigenschaft auf `CONVERT_TO_TEXT`.
- Um die Nutzlast ohne Änderung zu übergeben, lassen Sie die Eigenschaft undefiniert. Um eine binäre Nutzlast ohne Änderung zu übergeben, müssen Sie auch sicherstellen, dass der Content-Type mit einem der `binaryMediaTypes`-Einträge übereinstimmt und [Passthrough-Verhalten](#) für die API aktiviert sind.
- Legen Sie die `contentHandling` Eigenschaft der [IntegrationResponse](#) Ressource fest. Die `contentHandling`-Eigenschaft, der Accept-Header in Client-Anfragen und die `binaryMediaTypes` Ihrer APIs legen zusammen fest, wie API Gateway Inhaltstypkonvertierungen handhabt. Details hierzu finden Sie unter [the section called “Inhaltstypkonvertierungen in API Gateway”](#).

Important

Wenn eine Anfrage mehrere Medientypen in ihrem Accept-Header enthält, berücksichtigt API Gateway nur den ersten Accept-Medientyp. Wenn Sie die Reihenfolge der Accept-Medientypen nicht beeinflussen können und der Medientyp Ihres binären Inhalts nicht der erste in der Liste ist, fügen Sie den ersten Accept-Medientyp in der `binaryMediaTypes`-Liste Ihrer API hinzu. API Gateway behandelt alle Inhaltstypen in dieser Liste als binär. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.

Ausführliche Informationen darüber, wie API Gateway die Text- und Binär-Payloads behandelt, finden Sie unter [Inhaltstypkonvertierungen in API Gateway](#).

Inhaltstypkonvertierungen in API Gateway

Die Kombination aus den `binaryMediaTypes` Ihrer API, den Headern in Client-Anfragen und der Integrations-`contentHandling`-Eigenschaft bestimmt, wie API Gateway Payloads kodiert.

Die folgende Tabelle zeigt, wie API Gateway die Anforderungsnutzlast für bestimmte Konfigurationen des Content-Type Headers einer Anfrage, der `binaryMediaTypes` Liste einer [RestApi](#) Ressource und des `contentHandling` Eigenschaftswerts der [Integrationsressource](#) konvertiert.

Nutzlast der Methodenanforderung	Content-Type -Header der Anforderung	binaryMediaTypes	contentHandling	Nutzlast der Integrationsanforderung
Textdaten	Beliebiger Datentyp	Undefined	Undefined	UTF8-kodierte Zeichenfolge
Textdaten	Beliebiger Datentyp	Undefined	CONVERT_TO_BINARY	Base64-dekodierte binäre Blob
Textdaten	Beliebiger Datentyp	Undefined	CONVERT_TO_TEXT	UTF8-kodierte Zeichenfolge
Textdaten	Ein Textdatentyp	Satz mit übereinstimmenden Medientypen	Undefined	Textdaten
Textdaten	Ein Textdatentyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Base64-dekodierte binäre Blob
Textdaten	Ein Textdatentyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	Textdaten
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	Undefined	Binäre Daten
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Binäre Daten

Nutzlast der Methodenanforderung	Content-Type -Header der Anforderung	binaryMediaTypes	contentHandling	Nutzlast der Integrationsanforderung
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	Base64-kodierte Zeichenfolge

Die folgende Tabelle zeigt, wie API Gateway die Antwortnutzlast für bestimmte Konfigurationen des Accept Headers einer Anfrage, der `binaryMediaTypes` Liste einer [RestApi](#)Ressource und des `contentHandling` Eigenschaftswerts der [IntegrationResponse](#)Ressource konvertiert.

Important

Wenn eine Anfrage mehrere Medientypen in ihrem Accept-Header enthält, berücksichtigt API Gateway nur den ersten Accept-Medientyp. Wenn Sie die Reihenfolge der Accept-Medientypen nicht beeinflussen können und der Medientyp Ihres binären Inhalts nicht der erste in der Liste ist, fügen Sie den ersten Accept-Medientyp in der `binaryMediaTypes`-Liste Ihrer API hinzu. API Gateway behandelt alle Inhaltstypen in dieser Liste als binär. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.

Nutzlast der Integrationsantwort	Accept -Header der Anforderung	binaryMediaTypes	contentHandling	Nutzlast der Methodenantwort
Text- oder Binärdaten	Ein Texttyp	Undefined	Undefined	UTF8-kodierte Zeichenfolge
Text- oder Binärdaten	Ein Texttyp	Undefined	CONVERT_TO_BINARY	Base64-dekodierte Blob

Nutzlast der Integrationsantwort	Accept -Header der Anforderung	binaryMediaTypes	contentHandling	Nutzlast der Methodenantwort
Text- oder Binärdaten	Ein Texttyp	Undefined	CONVERT_TO_TEXT	UTF8-kodierte Zeichenfolge
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	Undefined	Textdaten
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Base64-dekodierte Blob
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	UTF8-kodierte Zeichenfolge
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	Undefined	Base64-dekodierte Blob
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Base64-dekodierte Blob
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	UTF8-kodierte Zeichenfolge

Nutzlast der Integrationsantwort	Accept -Header der Anforderung	binaryMediaTypes	contentHandling	Nutzlast der Methodenanantwort
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	Undefined	Base64-kodierte Zeichenfolge
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Binäre Daten
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	Base64-kodierte Zeichenfolge
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	Undefined	Binäre Daten
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_BINARY	Binäre Daten
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	CONVERT_TO_TEXT	Base64-kodierte Zeichenfolge

Bei Umwandlung einer Textnutzlast in ein binäres Blob geht API Gateway davon aus, dass es sich bei den Textdaten um eine base64-kodierte Zeichenfolge handelt, und gibt die Binärdaten als base64-dekodierten Blob zurück. Wenn die Konvertierung fehlschlägt, wird eine 500-Antwort

zurückgegeben, die auf einen API-Konfigurationsfehler hinweist. Für eine solche Konvertierung geben Sie keine Mapping-Vorlage an, auch wenn Sie das [Pass-Through-Verhalten](#) in der API aktivieren müssen.

Bei der Konvertierung eines binären Payloads in eine Textzeichenfolge wendet API Gateway immer eine base64-Kodierung auf die Binärdaten an. Sie können eine Mapping-Vorlage für eine solche Nutzlast definieren, aber nur auf die base64-kodierte Zeichenfolge in der Mapping-Vorlage durch `$input.body` zugreifen, wie im folgenden Beispielauszug einer Mapping-Vorlage dargestellt.

```
{
  "data": "$input.body"
}
```

Damit die binäre Nutzlast unverändert übergeben wird, müssen Sie das [Pass-Through-Verhalten](#) in der API aktivieren.

Binärunterstützung über die API Gateway-Konsole aktivieren

Der Abschnitt erklärt, wie die Binärunterstützung in der API Gateway-Konsole aktiviert wird. Als Beispiel verwenden wir eine API, die in Amazon S3 integriert ist. Wir konzentrieren uns darauf, die unterstützten Medientypen festzulegen und anzugeben, wie die Nutzlast verarbeitet werden soll. Ausführliche Informationen über die Erstellung einer in Amazon S3 integrierten API finden Sie unter [Tutorial: REST-API als AmazonS3-Proxy erstellen](#).

So aktivieren Sie Binärunterstützung mit Hilfe der API Gateway-Konsole:

1. Festlegen der binären Medientypen für die API:
 - a. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API aus. In diesem Beispiel nennen wir die API `FileMan`.
 - b. Klicken Sie unter der ausgewählten API im Hauptnavigationsbereich auf API-Einstellungen.
 - c. Im Bereich API-Einstellungen klicken Sie dann im Abschnitt Binäre Medientypen verwalten auf Binären Medientyp hinzufügen.
 - d. Klicken Sie auf Binären Medientyp hinzufügen.
 - e. Geben Sie einen erforderlichen Medientyp (z. B. **image/png**) im Eingabetextfeld ein. Wiederholen Sie diesen Schritt bei Bedarf, um weitere Medientypen hinzuzufügen. Um alle binären Medientypen zu unterstützen, legen Sie fest `*/*`.
 - f. Wählen Sie **Änderungen speichern** aus.

2. Legen Sie fest, wie Nachrichtennutzlasten für die API-Methode behandelt werden sollen:
 - a. Erstellen Sie eine neue Ressource oder wählen Sie eine vorhandene Ressource in der API aus. In diesem Beispiel wird die Ressource `/ {folder} / {item}` verwendet.
 - b. Erstellen Sie eine neue Methode oder wählen Sie eine vorhandene Methode für die Ressource aus. Als Beispiel verwenden wir die `GET / {folder} / {item}`-Methode, die mit der `Object GET`-Aktion in Amazon S3 integriert ist.
 - c. Wählen Sie unter Inhaltsbehandlung eine Option aus.

Action type

☐ Use action name

☒ Use path override

Path override

`{bucket}/{object}`

Execution role

`arn:aws:iam::444455556666:role/s3-ApiGatewayS3ReadOnlyRole`

Credential cache

Do not add caller credentials to cache key ▼

Content handling | [Learn more](#)

Passthrough ▼

Wählen Sie Pass-Through, wenn der Textkörper nicht konvertiert werden soll, wenn Client und Backend das gleiche Binärformat akzeptieren. Klicken Sie auf **In Text umwandeln**, um den binären Text in eine base64-kodierte Zeichenfolge zu konvertieren, wenn beispielsweise eine binäre Anforderungsnutzlast als JSON-Eigenschaft für das Backend erforderlich ist. Klicken Sie auf **In Binärdaten umwandeln**, wenn der Client eine base64-kodierte Zeichenfolge übermittelt, aber für das Backend das ursprüngliche Binärformat erforderlich ist, oder wenn der Endpunkt eine base64-kodierte Zeichenfolge zurückgibt und der Client nur eine binäre Ausgabe akzeptiert.

- d. Wählen Sie für Anforderungstext-Pass-Through die Option **Wenn keine Vorlagen definiert sind** (empfohlen) aus.

Alternativ können Sie auch **Nie** auswählen. Dies bedeutet, dass die API Daten mit Inhaltstypen ablehnt, die keine Zuweisungsvorlage enthalten.

- e. Behalten Sie den Accept-Header der eingehenden Anforderung in der Integrationsanforderung bei. Sie sollten dies auch tun, wenn Sie `contentHandling` auf `passthrough` festgelegt haben und diese Einstellung zur Laufzeit überschrieben werden soll.

HTTP headers (2)			< 1 >
Name	Mapped from	Caching	
Accept	method.request.header.Accept	☐ Inactive	
Content-Type	method.request.header.Content-Type	☐ Inactive	

- f. Für eine Umwandlung in Text definieren Sie eine Mapping-Vorlage, um die base64-kodierten binären Daten in das erforderliche Format zu bringen.

Im Folgenden sehen Sie ein Beispiel für eine Zuordnungsvorlage, die in Text konvertiert werden kann:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

Das Format dieser Mapping-Vorlage hängt von den Anforderungen des Endpunkts an die Eingabe ab.

- g. Wählen Sie Speichern.

Binärunterstützung über die API Gateway-REST-API aktivieren

Die folgenden Aufgaben zeigen, wie die Binärunterstützung unter Verwendung der API Gateway-REST-API-Aufrufe aktiviert werden kann.

Themen

- [Hinzufügen und Aktualisieren der unterstützten binären Medientypen für eine API](#)
- [Konfigurieren der Konvertierung von Anforderungsnutzlasten](#)
- [Konfigurieren der Konvertierung von Antwortnutzlasten](#)
- [Konvertieren von Binärdaten in Textdaten](#)
- [Konvertieren von Textdaten in eine binäre Nutzlast](#)
- [Übergeben einer binären Nutzlast](#)

Hinzufügen und Aktualisieren der unterstützten binären Medientypen für eine API

Um API Gateway die Unterstützung eines neuen binären Medientyps zu ermöglichen, müssen Sie den binären Medientyp zur `binaryMediaTypes`-Liste der `RestApi`-Ressource hinzufügen. Wenn API Gateway beispielsweise JPEG-Bilder verarbeiten soll, müssen Sie eine PATCH-Anfrage an die `RestApi`-Ressource stellen:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
]
```

Die MIME-Typ-Spezifikation `image/jpeg`, die Teil des `path`-Eigenschaftswerts ist, wird als `image~1jpeg` kodiert.

Um die unterstützten binären Medientypen zu aktualisieren, ersetzen oder entfernen Sie sie aus der Liste `binaryMediaTypes` der `RestApi`-Ressource. Wenn Sie beispielsweise die Binärunterstützung von JPEG-Dateien in Rohbytes ändern möchten, senden Sie wie folgt eine PATCH-Anforderung an die `RestApi`-Ressource:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
]
```

Konfigurieren der Konvertierung von Anforderungsnutzlasten

Wenn der Endpunkt eine binäre Eingabe erfordert, legen Sie die `contentHandling`-Eigenschaft der `Integration-Ressource` auf `CONVERT_TO_BINARY` fest. Um dies zu tun, senden Sie eine `PATCH`-Anforderung wie folgt:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  } ]
}
```

Konfigurieren der Konvertierung von Antwortnutzlasten

Wenn der Client das Ergebnis als binäres Blob anstelle der base64-kodierten Nutzlast, die vom Endpunkt zurückgegeben wird, akzeptiert, legen Sie die `contentHandling`-Eigenschaft für die `IntegrationResponse-Ressource` auf `CONVERT_TO_BINARY` fest. Um dies zu tun, senden Sie eine `PATCH`-Anforderung wie folgt:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  } ]
}
```

Konvertieren von Binärdaten in Textdaten

Gehen Sie wie folgt vor, um Binärdaten als JSON-Eigenschaft der Eingabe an AWS Lambda oder Kinesis über API Gateway zu senden:

1. Aktivieren Sie die Unterstützung binärer Nutzlasten der API durch Hinzufügen des neuen binären Medientyps `application/octet-stream` zur Liste `binaryMediaTypes` der API.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
]
}
```

2. Legen Sie `CONVERT_TO_TEXT` für die Eigenschaft `contentHandling` der Integration-Ressource fest und geben Sie eine Mapping-Vorlage an, um die base64-kodierte Zeichenfolge der Binärdaten einer JSON-Eigenschaft zuzuweisen. Im folgenden Beispiel ist die JSON-Eigenschaft `body` und `$input.body` enthält die base64-kodierte Zeichenfolge.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_TEXT"
    },
    {
      "op" : "add",
      "path" : "/requestTemplates/application~1octet-stream",
      "value" : "{\"body\": \"$input.body\"}"
    }
  ]
}
```

Konvertieren von Textdaten in eine binäre Nutzlast

Angenommen, eine Lambda-Funktion gibt eine Bilddatei als base64-kodierte Zeichenfolge zurück. Gehen Sie wie folgt vor, um diese Binärausgabe über API Gateway an den Client zu übergeben:

1. Aktualisieren Sie die Liste `binaryMediaTypes` der API durch Hinzufügen des binären Medientyps `application/octet-stream`, sofern dieser noch nicht in der Liste enthalten ist.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Legen Sie die `contentHandling`-Eigenschaft für die Integration-Ressource auf `CONVERT_TO_BINARY` fest. Definieren Sie keine Mapping-Vorlage. Wenn Sie keine Mapping-Vorlage definieren, ruft API Gateway die Passthrough-Vorlage auf, um den base64-dekodierten binären Blob als Bilddatei an den Client zurückzugeben.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Übergeben einer binären Nutzlast

Um ein Bild in einem Amazon S3-Bucket mit API Gateway zu speichern, gehen Sie wie folgt vor:

1. Aktualisieren Sie die Liste `binaryMediaTypes` der API durch Hinzufügen des binären Medientyps `application/octet-stream`, sofern dieser noch nicht in der Liste enthalten ist.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
```

```

    }
  ]
}

```

2. Legen Sie die `contentHandling`-Eigenschaft für die `Integration`-Ressource auf `CONVERT_TO_BINARY` fest. Legen Sie `WHEN_NO_MATCH` als `passthroughBehavior`-Eigenschaftswert fest, ohne eine Mapping-Vorlage zu definieren. Dadurch wird API Gateway in die Lage versetzt, die Passthrough-Vorlage aufzurufen.

```

PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    },
    {
      "op" : "replace",
      "path" : "/passthroughBehaviors",
      "value" : "WHEN_NO_MATCH"
    }
  ]
}

```

API Gateway-Importe und Exporte von Inhaltskodierungen

Um die `binaryMediaTypes` Liste in eine zu importieren [RestApi](#), verwenden Sie die folgende API Gateway Gateway-Erweiterung zur OpenAPI-Definitionsdatei der API. Die Erweiterung wird auch verwendet, um die API-Einstellungen zu exportieren.

- [x-amazon-apigateway-binaryEigenschaft -media-types](#)

Um den `contentHandling`-Eigenschaftswert für eine `Integration`- oder `IntegrationResponse`-Ressource zu importieren und zu exportieren, verwenden Sie die folgenden API Gateway-Erweiterungen für die OpenAPI-Definitionen:

- [x-amazon-apigateway-integration Objekt](#)

- [x-amazon-apigateway-integration.response-Objekt](#)

Rückgabe binärer Medien aus einer Lambda-Proxy-Integration in API Gateway

Zum Zurückgeben binärer Medien aus einer [AWS Lambda -Proxy-Integration](#) verwenden Sie für die Antwort Ihrer Lambda-Funktion base64-Codierung. Sie müssen auch [die binären Medientypen Ihrer API konfigurieren](#). Wenn Sie die binären Medientypen Ihrer API konfigurieren, behandelt Ihre API diesen Inhaltstyp als binäre Daten. Die maximale Größe der Nutzlast ist 10 MB.

Note

Um einen Webbrowser zum Aufruf einer API mit dieser Beispielintegration zu verwenden, legen Sie die binären Medientypen Ihrer API auf `*/*`. API Gateway verwendet den ersten Accept-Header von Clients, um zu bestimmen, ob eine Antwort binäre Medien zurückgeben soll. Um binäre Medien zurückzugeben, wenn Sie die Reihenfolge der Accept-Headerwerte nicht steuern können, z. B. Anforderungen von einem Browser, setzen Sie die binären Medientypen der API (für alle Inhaltstypen) auf `*/*`.

Die folgende Beispiel-Lambda-Funktion kann ein binäres Bild von Amazon S3 oder Text an Clients zurückgeben. Die Antwort der Funktion enthält einen Content-Type-Header, der dem Client den Typ der zurückgegebenen Daten angibt. Die Funktion legt die `isBase64Encoded`-Eigenschaft bedingt in ihrer Antwort fest, abhängig von der Art der Daten, die sie zurückgibt.

Node.js

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3"

const client = new S3Client({region: 'us-east-2'});

export const handler = async (event) => {

  var randomint = function(max) {
    return Math.floor(Math.random() * max);
  }
  var number = randomint(2);
  if (number == 1){
    const input = {
      "Bucket" : "bucket-name",
      "Key" : "image.png"
```



```

    }
    try {
        const command = new GetObjectCommand(input)
        const response = await client.send(command);
        var str = await response.Body.transformToByteArray();
    } catch (err) {
        console.error(err);
    }
    const base64body = Buffer.from(str).toString('base64');
    return {
        'headers': { "Content-Type": "image/png" },
        'statusCode': 200,
        'body': base64body,
        'isBase64Encoded': true
    }
} else {
    return {
        'headers': { "Content-Type": "text/html" },
        'statusCode': 200,
        'body': "<h1>This is text</h1>",
    }
}
}

```

Python

```

import base64
import boto3
import json
import random

s3 = boto3.client('s3')

def lambda_handler(event, context):
    number = random.randint(0,1)
    if number == 1:
        response = s3.get_object(
            Bucket='bucket-name',
            Key='image.png',
        )
        image = response['Body'].read()
        return {
            'headers': { "Content-Type": "image/png" },

```

```
        'statusCode': 200,  
        'body': base64.b64encode(image).decode('utf-8'),  
        'isBase64Encoded': True  
    }  
    else:  
        return {  
            'headers': { "Content-type": "text/html" },  
            'statusCode': 200,  
            'body': "<h1>This is text</h1>",  
        }
```

Weitere Informationen zu binären Medientypen finden Sie unter [Binäre Medientypen für REST APIs in API Gateway](#).

Über eine API Gateway-API auf Binärdateien in Amazon S3 zugreifen

Die folgenden Beispiele zeigen die OpenAPI-Datei, die für den Zugriff auf Bilder in Amazon S3 verwendet wird, wie man ein Bild von Amazon S3 herunterlädt und wie man ein Bild in Amazon S3 hochlädt.

Themen

- [OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Amazon S3](#)
- [Bilddatei von Amazon S3 herunterladen](#)
- [Bilde in Amazon S3 hochladen](#)

OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Amazon S3

Die folgende OpenAPI-Datei zeigt eine Beispiel-API, die das Herunterladen einer Bilddatei von Amazon S3 und das Hochladen einer Bilddatei in Amazon S3 illustriert. Diese API stellt die Methoden GET /s3?key={file-name} und PUT /s3?key={file-name} zum Herunter- und Hochladen einer bestimmten Bilddatei bereit. Die Methode GET gibt die Bilddatei als base64-kodierte Zeichenfolge als Teil einer JSON-Ausgabe zurück, und zwar nach der bereitgestellten Mapping-Vorlage, in einer Antwort "200 OK". Die Methode PUT übernimmt einen unformatierten binären Blob als Eingabe und gibt eine Antwort "200 OK" mit einer leeren Nutzlast zurück.

OpenAPI 3.0

```
{  
  "openapi": "3.0.0",
```

```
"info": {
  "version": "2016-10-21T17:26:28Z",
  "title": "ApiName"
},
"paths": {
  "/s3": {
    "get": {
      "parameters": [
        {
          "name": "key",
          "in": "query",
          "required": false,
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
              }
            }
          }
        },
        "500": {
          "description": "500 response"
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
      "responses": {
        "default": {
          "statusCode": "500"
        },
        "2\\d{2}": {
          "statusCode": "200"
        }
      },
      "requestParameters": {
        "integration.request.path.key": "method.request.querystring.key"
      }
    }
  }
}
```

```

        },
        "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
        "passthroughBehavior": "when_no_match",
        "httpMethod": "GET",
        "type": "aws"
    }
},
"put": {
    "parameters": [
        {
            "name": "key",
            "in": "query",
            "required": false,
            "schema": {
                "type": "string"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "content": {
                "application/json": {
                    "schema": {
                        "$ref": "#/components/schemas/Empty"
                    }
                },
                "application/octet-stream": {
                    "schema": {
                        "$ref": "#/components/schemas/Empty"
                    }
                }
            }
        },
        "500": {
            "description": "500 response"
        }
    },
    "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
        "responses": {
            "default": {
                "statusCode": "500"
            }
        },

```

```

        "2\\d{2}": {
            "statusCode": "200"
        },
        "requestParameters": {
            "integration.request.path.key": "method.request.querystring.key"
        },
        "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
        "passthroughBehavior": "when_no_match",
        "httpMethod": "PUT",
        "type": "aws",
        "contentHandling": "CONVERT_TO_BINARY"
    }
}
},
"x-amazon-apigateway-binary-media-types": [
    "application/octet-stream",
    "image/jpeg"
],
"servers": [
    {
        "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
        "variables": {
            "basePath": {
                "default": "/v1"
            }
        }
    }
],
"components": {
    "schemas": {
        "Empty": {
            "type": "object",
            "title": "Empty Schema"
        }
    }
}
}

```

OpenAPI 2.0

```
{
```

```
"swagger": "2.0",
"info": {
  "version": "2016-10-21T17:26:28Z",
  "title": "ApiName"
},
"host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
"basePath": "/v1",
"schemes": [
  "https"
],
"paths": {
  "/s3": {
    "get": {
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "key",
          "in": "query",
          "required": false,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          }
        },
        "500": {
          "description": "500 response"
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
      "responses": {
        "default": {
          "statusCode": "500"
        },
        "2\\d{2}": {
          "statusCode": "200"
        }
      }
    }
  },
}
```

```
    "requestParameters": {
      "integration.request.path.key": "method.request.querystring.key"
    },
    "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "GET",
    "type": "aws"
  }
},
"put": {
  "produces": [
    "application/json", "application/octet-stream"
  ],
  "parameters": [
    {
      "name": "key",
      "in": "query",
      "required": false,
      "type": "string"
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Empty"
      }
    },
    "500": {
      "description": "500 response"
    }
  },
  "x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
    "responses": {
      "default": {
        "statusCode": "500"
      },
      "2\\d{2}": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.path.key": "method.request.querystring.key"
    }
  }
}
```

```
    },
    "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "PUT",
    "type": "aws",
    "contentHandling" : "CONVERT_TO_BINARY"
  }
}
},
"x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
"definitions": {
  "Empty": {
    "type": "object",
    "title": "Empty Schema"
  }
}
}
```

Bilddatei von Amazon S3 herunterladen

So laden Sie eine Bilddatei (`image.jpg`) als binären Blob von Amazon S3 herunter:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Die erfolgreiche Antwort schaut wie folgt aus:

```
200 OK HTTP/1.1

[raw bytes]
```

Die Rohbytes werden zurückgegeben, da der Accept-Header auf den binären Medientyp `application/octet-stream` und die Binärunterstützung für die API aktiviert ist.

Eine Alternative besteht darin, die Bilddatei (`image.jpg`) als eine base64-kodierte Zeichenfolge (formatiert als JSON-Eigenschaft) von Amazon S3 herunterzuladen. Fügen Sie der 200-

Integrationsantwort eine Antwortvorlage hinzu, wie im folgenden fettgedruckten OpenAPI-Definitionsblock gezeigt:

```
"x-amazon-apigateway-integration": {
  "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
  "responses": {
    "default": {
      "statusCode": "500"
    },
    "2\\d{2}": {
      "statusCode": "200",
      "responseTemplates": {
        "application/json": "{\n  \"image\": \"$input.body\"\n}"
      }
    }
  },
},
```

Die Anforderung zum Herunterladen der Bilddatei sieht folgendermaßen aus:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

{
  "image": "W3JhdYBieXRlc10="
}
```

Bilde in Amazon S3 hochladen

So laden Sie eine Bilddatei (image.jpg) als binären Blob in Amazon S3 hoch:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json
```

```
[raw bytes]
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge in Amazon S3 hoch:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Die Eingabenutzlast muss eine base64-kodierte Zeichenfolge sein, da der Content-Type-Headerwert auf `application/json` festgelegt ist. Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1
```

Über eine API Gateway-API auf Binärdateien in Lambda zugreifen

Das folgende OpenAPI-Beispiel zeigt, wie Sie AWS Lambda über eine API-Gateway-API auf eine Binärdatei zugreifen. Diese API legt die Methoden `GET /lambda?key={file-name}` und `PUT /lambda?key={file-name}` zum Herunter- und Hochladen einer bestimmten Bilddatei offen. Die Methode `GET` gibt die Bilddatei als base64-kodierte Zeichenfolge als Teil einer JSON-Ausgabe zurück, und zwar nach der bereitgestellten Mapping-Vorlage, in einer Antwort "200 OK". Die Methode `PUT` übernimmt einen unformatierten binären Blob als Eingabe und gibt eine Antwort "200 OK" mit einer leeren Nutzlast zurück.

Sie erstellen die Lambda-Funktion, die von Ihrer API aufgerufen wird. Diese muss eine Base64-kodierte Zeichenfolge mit dem Header Content-Type von `application/json` zurückgeben.

Themen

- [OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Lambda](#)
- [Bilddatei von Lambda herunterladen](#)
- [Bild in Lambda hochladen](#)

OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Lambda

Die folgende OpenAPI-Datei zeigt eine Beispiel-API, die das Herunterladen einer Bilddatei von Lambda und das Hochladen einer Bilddatei in Lambda veranschaulicht.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "paths": {
    "/lambda": {
      "get": {
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
```

```

        "type": "AWS",
        "credentials": "arn:aws:iam::123456789012:role/Lambda",
        "httpMethod": "POST",
        "requestTemplates": {
            "application/json": "{\n  \"imageKey\":\n\n\"$input.params('key')\"\n}"
        },
        "responses": {
            "default": {
                "statusCode": "500"
            },
            "2\\d{2}": {
                "statusCode": "200",
                "responseTemplates": {
                    "application/json": "{\n  \"image\": \"$input.body\"\n}"
                }
            }
        }
    },
    "put": {
        "parameters": [
            {
                "name": "key",
                "in": "query",
                "required": false,
                "schema": {
                    "type": "string"
                }
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "content": {
                    "application/json": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    },
                    "application/octet-stream": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    },
    "500": {
        "description": "500 response"
    }
},
"x-amazon-apigateway-integration": {
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
    "type": "AWS",
    "credentials": "arn:aws:iam::123456789012:role/Lambda",
    "httpMethod": "POST",
    "contentHandling": "CONVERT_TO_TEXT",
    "requestTemplates": {
        "application/json": "{\n    \"imageKey\": \"${input.params('key')}\",
    \"image\": \"${input.body}\"}\n}"
    },
    "responses": {
        "default": {
            "statusCode": "500"
        },
        "2\\d{2}": {
            "statusCode": "200"
        }
    }
}
}
},
"x-amazon-apigateway-binary-media-types": [
    "application/octet-stream",
    "image/jpeg"
],
"servers": [
    {
        "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
        "variables": {
            "basePath": {
                "default": "/v1"
            }
        }
    }
]
],

```

```
    "components": {
      "schemas": {
        "Empty": {
          "type": "object",
          "title": "Empty Schema"
        }
      }
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      }
    }
  }
}
```

```

    },
    "500": {
      "description": "500 response"
    }
  },
  "x-amazon-apigateway-integration": {
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
    "type": "AWS",
    "credentials": "arn:aws:iam::123456789012:role/Lambda",
    "httpMethod": "POST",
    "requestTemplates": {
      "application/json": "{\n  \"imageKey\": \"${input.params('key')}\n}"
    },
    "responses": {
      "default": {
        "statusCode": "500"
      },
      "2\\d{2}": {
        "statusCode": "200",
        "responseTemplates": {
          "application/json": "{\n  \"image\": \"${input.body}\n}"
        }
      }
    }
  }
},
"put": {
  "produces": [
    "application/json", "application/octet-stream"
  ],
  "parameters": [
    {
      "name": "key",
      "in": "query",
      "required": false,
      "type": "string"
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Empty"
      }
    }
  }
}

```

```

    }
  },
  "500": {
    "description": "500 response"
  }
},
"x-amazon-apigateway-integration": {
  "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
  "type": "AWS",
  "credentials": "arn:aws:iam::123456789012:role/Lambda",
  "httpMethod": "POST",
  "contentHandling" : "CONVERT_TO_TEXT",
  "requestTemplates": {
    "application/json": "{\n  \"imageKey\": \"${input.params('key')}\",
    \"image\": \"${input.body}\"}\n}"
  },
  "responses": {
    "default": {
      "statusCode": "500"
    },
    "2\\d{2}": {
      "statusCode": "200"
    }
  }
}
}
}
},
"x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
"definitions": {
  "Empty": {
    "type": "object",
    "title": "Empty Schema"
  }
}
}
}

```

Bilddatei von Lambda herunterladen

So laden Sie eine Bilddatei (`image.jpg`) als binäres Blob von Lambda herunter:


```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

[raw bytes]
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge, formatiert als JSON-Eigenschaft, von Lambda herunter:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Bild in Lambda hochladen

So laden Sie eine Bilddatei (`image.jpg`) als binäres Blob in Lambda hoch:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge in Lambda hoch:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK
```

Aufrufen von REST-APIs in API Gateway

Um eine bereitgestellte API aufzurufen, senden Kunden Anfragen an die URL für den API Gateway-Komponenten-Service zur API-Ausführung (bekannt als `execute-api`).

Die Basis-URL für REST-API liegt im folgenden Format vor:

```
https://api-id.execute-api.region.amazonaws.com/stage/
```

wobei *api-id* die API-Kennung, *region* die AWS-Region und *stage* der Stufenname der API-Bereitstellung ist.

Important

Bevor Sie eine API aufrufen können, müssen Sie sie in API Gateway bereitstellen. Anleitungen zur Bereitstellung einer API finden Sie unter [REST-APIs in API Gateway bereitstellen](#).

Themen

- [Die Aufruf-URL einer API abrufen](#)
- [Aufrufen einer API](#)

- [API Gateway-Konsole zum Testen einer REST-API-Methode verwenden](#)
- [Von API Gateway generiertes Java-SDK für eine REST-API verwenden](#)
- [Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden](#)
- [Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API](#)
- [Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden](#)
- [Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden](#)

Die Aufruf-URL einer API abrufen

Sie können die Konsole, AWS CLI oder eine exportierte OpenAPI-Definition verwenden, um die Aufruf-URL einer API abzurufen.

Die Aufruf-URL einer API in der Konsole abrufen

Im folgenden Verfahren wird gezeigt, wie Sie die Aufruf-URL einer API in der REST-API-Konsole abrufen.

Abrufen der Aufruf-URL einer API in der REST-API-Konsole

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine bereitgestellte API aus.
3. Wählen Sie im Haupt-Navigationsbereich Stufe aus.
4. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

Diese URL ist für die Root-Ressource Ihrer API.

Stage details [Info](#) [Edit](#)

Stage name Prod	Rate Info 10000	Web ACL -
Cache cluster Info ⊖ Inactive	Burst Info 5000	Client certificate -
Default method-level caching ⊖ Inactive		

Invoke URL
 `https://abcd1234.execute-api.us-east-1.amazonaws.com/Prod`

Active deployment
efg456 on June 01, 2023, 15:00 (UTC-07:00)

5. Vergrößern Sie die Stufe im sekundären Navigationsbereich und wählen Sie dann eine Methode aus, um die Aufruf-URL einer API für eine andere Ressource in Ihrer API abzurufen.
6. Klicken Sie auf das Kopiersymbol, um die Aufruf-URL Ihrer API auf Ressourcenebene zu kopieren.

Stages

Stage actions ▼

Create stage

prod

/

GET

/pets

GET

OPTIONS

POST

/{petId}

GET

OPTIONS

Method overrides

Edit

By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

 This method inherits its settings from the 'prod' stage.

Invoke URL

 <https://abcd1234.execute-api.us-west-1.amazonaws.com/prod/pets>

Aufruf-URL einer API mithilfe der AWS CLI abrufen

Im folgenden Verfahren wird gezeigt, wie Sie die Aufruf-URL einer API mithilfe der AWS CLI abrufen.

Aufruf-URL einer API mithilfe der AWS CLI abrufen

1. Rufen Sie die `rest-api-id` mit folgendem Befehl ab. Dieser Befehl gibt alle `rest-api-id`-Werte in Ihrer Region zurück. Weitere Informationen finden Sie unter [get-rest-apis](#).

```
aws apigateway get-rest-apis
```

2. Ersetzen Sie die Beispiel-rest-api-id durch Ihre rest-api-id, den Beispiel-`{stage-name}` durch Ihren `{stage-name}` und die `{region}` durch Ihre Region.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Abrufen der Aufruf-URL einer API mithilfe der exportierten OpenAPI-Definitionsdatei der API

Sie können die Root-URL auch zusammensetzen, indem Sie die Felder host und basePath einer exportierten OpenAPI-Definitionsdatei der API kombinieren. Eine Anleitung zum Exportieren Ihrer API finden Sie unter [the section called “Exportieren einer REST-API”](#).

Aufrufen einer API

Sie können Ihre bereitgestellte API im Browser, mit Curl oder anderen Anwendungen wie [Postman](#) aufrufen.

Darüber hinaus können Sie einen API-Aufruf mithilfe der API-Gateway-Konsole testen. Der Test verwendet die TestInvoke-Funktion des API-Gateways, die API-Tests vor der Bereitstellung der API ermöglicht. Weitere Informationen finden Sie unter [the section called “Testen einer REST-API-Methode mithilfe der Konsole”](#).

Note

In Parameterwerten für Abfragezeichenfolgen in einer Aufruf-URL ist %% nicht zulässig.

Aufrufen einer API mit einem Webbrowser

Wenn Ihre API anonyme Zugriffe gestattet, können Sie jeden Webbrowser verwenden, um eine beliebige GET-Methode aufzurufen. Geben Sie die vollständige Aufruf-URL in die Adressleiste des Browsers ein.

Bei anderen Methoden oder Aufrufen, für die eine Authentifizierung erforderlich ist, müssen Sie eine Nutzlast angeben oder die Anforderungen signieren. Sie können diese mithilfe eines der AWS-SDKs in einem Skript hinter einer HTML-Seite oder in einer Client-Anwendung verarbeiten.

Aufrufen einer API mit Curl

Sie können ein Tool wie [Curl](#) in Ihrem Terminal verwenden, um Ihre API aufzurufen. Der folgende CURL-Beispielbefehl ruft die GET-Methode für die getUsers-Ressource in der prod-Stufe einer API auf.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers"
```

API Gateway-Konsole zum Testen einer REST-API-Methode verwenden

API Gateway-Konsole zum Testen einer REST-API-Methode verwenden

Themen

- [Voraussetzungen](#)
- [Methode mit der API Gateway-Konsole testen](#)

Voraussetzungen

- Sie müssen die Einstellungen für die Methoden angeben, die Sie testen möchten. Folgen Sie den Anweisungen in [Methoden für REST APIs in API Gateway](#).

Methode mit der API Gateway-Konsole testen

Important

Das Testen von Methoden mit der API Gateway-Konsole kann zu Änderungen an Ressourcen führen, die nicht rückgängig gemacht werden können. Das Testen einer Methode mit der API Gateway-Konsole ist dasselbe wie der Aufruf der Methode außerhalb der API Gateway-Konsole. Wenn Sie beispielsweise über die Amazon API Gateway-Konsole eine Methode aufrufen, die die Ressourcen einer API löscht, werden diese API-Ressourcen bei einem erfolgreichen Methodenaufruf gelöscht.

So testen Sie eine Methode

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Bereich Resources auf die Methode, die Sie testen möchten.
4. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

Method request | Integration request | Integration response | Method response | **Test**

Test method
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Query strings
dog=2

Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2

Client certificate
No client certificates have been generated.

Test

Geben Sie Werte in eines der angezeigten Felder ein (z. B. Abfragezeichenfolgen, Header und Anforderungstext). Die Konsole schließt diese Werte im Standard-Anwendungs-/JSON-Format in die Methodenanforderung ein.

Wenn Sie weitere Optionen angeben müssen, wenden Sie sich an den API-Eigentümer.

5. Wählen Sie Test aus. Sie bekommen die folgenden Informationen angezeigt:
 - Request ist der Pfad der Ressource, die für die Methode aufgerufen wurde.
 - Status ist der HTTP-Statuscode der Antwort.
 - Latenz in ms ist die Zeit zwischen dem Empfang der Anforderung vom Aufrufer und der Antwortrückgabe.
 - Antworttext ist der Text der HTTP-Antwort.

- Antwort-Header sind die Header der HTTP-Antwort.

 Tip

Abhängig von dem Mapping können der HTTP-Statuscode, Antworttext und die Antwort-Header von denen abweichen, die von der Lambda-Funktion, dem HTTP-Proxy oder dem AWS-Service-Proxy gesendet werden.

- Protokolle sind die simulierten Einträge in Amazon CloudWatch Logs, die geschrieben worden wären, wenn diese Methode außerhalb der API Gateway-Konsole aufgerufen worden wäre.

 Note

Obwohl die CloudWatch Logs-Einträge simuliert werden, sind die Ergebnisse des Methodenaufrufs echt.

Zusätzlich zur Verwendung der API-Gateway-Konsole können Sie die AWS CLI oder ein AWS-SDK für API Gateway verwenden, um den Aufruf einer Methode zu testen. Die Vorgehensweise mithilfe der AWS CLI wird unter [test-invoke-method](#) beschrieben.

Von API Gateway generiertes Java-SDK für eine REST-API verwenden

In diesem Abschnitt werden die Schritte zur Verwendung eines von API Gateway generierten Java-SDKs für eine REST-API unter Verwendung der [Simple Calculator](#)-API als Beispiel vorgestellt. Bevor Sie fortfahren, müssen Sie die in [Generieren Sie SDKs für REST APIs in API Gateway](#) beschriebenen Schritte durchführen.

So installieren und verwenden Sie ein von API Gateway generiertes Java-SDK:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.
2. Laden Sie [Apache Maven](#) (Version 3.5 oder höher) herunter, und installieren Sie es.
3. Laden Sie [JDK 8](#) herunter und installieren Sie die Software.
4. Legen Sie die JAVA_HOME-Umgebungsvariable fest.
5. Öffnen Sie den entpackten SDK-Ordner, in dem die Datei pom.xml gespeichert ist. Dieser Ordner heißt standardmäßig generated-code. Führen Sie den Befehl `mvn install` aus, um die

kompilierten Artefaktdateien in Ihrem lokalen Maven-Repository zu installieren. Dadurch wird ein Ordner namens `target` erzeugt, der die kompilierte SDK-Bibliothek enthält.

6. Geben Sie den folgenden Befehl in einem leeren Verzeichnis ein, um ein Client-Projekt-Stub zu erstellen, um die API über die installierte SDK-Bibliothek aufzurufen.

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
  -DartifactId=SimpleCalc-sdkClient
```

Note

Das Trennzeichen `\` in dem vorausgehenden Befehl dient der besseren Lesbarkeit. Der gesamte Befehl sollte in einer einzelnen Zeile ohne das Trennzeichen stehen.

Mit diesem Befehl wird ein Anwendungs-Stub erstellt. Das Anwendungs-Stub enthält die Datei `pom.xml` und einen `src`-Ordner im Stammverzeichnis des Projekts (*SimpleCalc sdkClient* im vorausgehenden Befehl). Anfänglich gibt es zwei Quelldateien: `src/main/java/{package-path}/App.java` und `src/test/java/{package-path}/AppTest.java`. In diesem Beispiel ist *{package-path}* `examples/aws/apig/simpleCalc/sdk/app`. Dieser Paketpfad ist von dem Wert `DarchetypeGroupId` abgeleitet. Sie können die Datei `App.java` als Vorlage für Ihre Client-Anwendung verwenden, und Sie können nach Bedarf weitere in demselben Ordner hinzufügen. Sie können die Datei `AppTest.java` als Testvorlage für Ihre Anwendung benutzen, und Sie können nach Bedarf weitere Testcode-Dateien in demselben Ordner hinzufügen.

7. Aktualisieren Sie die Paket-Abhängigkeiten in der generierten Datei `pom.xml` wie folgt; ersetzen Sie dabei nötigenfalls die Eigenschaften `groupId`, `artifactId`, `version` und `name` Ihres Projekts:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/
POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>examples.aws.apig.simpleCalc.sdk.app</groupId>
  <artifactId>SimpleCalc-sdkClient</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
```

```
<name>SimpleCalc-sdkClient</name>
<url>http://maven.apache.org</url>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-core</artifactId>
    <version>1.11.94</version>
  </dependency>
  <dependency>
    <groupId>my-apig-api-examples</groupId>
    <artifactId>simple-calc-sdk</artifactId>
    <version>1.0.0</version>
  </dependency>

  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.5</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.5.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

 Note

Wenn eine neuere Version des abhängigen Artefakts von `aws-java-sdk-core` nicht mit der Version des oben angegebenen (1.11.94) kompatibel ist, müssen Sie das `<version>`-Tag auf die neue Version aktualisieren.

8. Als Nächstes zeigen wir, wie Sie die API mithilfe des SDK aufrufen, indem Sie die SDK-Methoden `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` und `postApiRoot(PostApiRootRequest req)` aufrufen. Diese Methoden entsprechen den Methoden `GET /{a}/{b}/{op}`, `GET /?a={x}&b={y}&op={operator}` und `POST /`, mit einer jeweiligen Nutzlast von `{"a": x, "b": y, "op": "operator"}`-API-Anforderungen.

Aktualisieren Sie die Datei `App.java` wie folgt:

```
package examples.aws.apig.simpleCalc.sdk.app;

import java.io.IOException;

import com.amazonaws.opensdk.config.ConnectionConfiguration;
import com.amazonaws.opensdk.config.TimeoutConfiguration;

import examples.aws.apig.simpleCalc.sdk.*;
import examples.aws.apig.simpleCalc.sdk.model.*;
import examples.aws.apig.simpleCalc.sdk.SimpleCalcSdk.*;

public class App
{
    SimpleCalcSdk sdkClient;

    public App() {
        initSdk();
    }

    // The configuration settings are for illustration purposes and may not be a
    // recommended best practice.
    private void initSdk() {
        sdkClient = SimpleCalcSdk.builder()
            .connectionConfiguration(
                new ConnectionConfiguration()
                    .maxConnections(100)
                    .connectionMaxIdleMillis(1000))
    }
```

```

        .timeoutConfiguration(
            new TimeoutConfiguration()
                .httpRequestTimeout(3000)
                .totalExecutionTimeout(10000)
                .socketTimeout(2000))
        .build();
    }
    // Calling shutdown is not necessary unless you want to exert explicit control
    of this resource.
    public void shutdown() {
        sdkClient.shutdown();
    }

    // GetABOpResult getABOp(GetABOpRequest getABOpRequest)
    public Output getResultWithPathParameters(String x, String y, String operator)
    {
        operator = operator.equals("+") ? "add" : operator;
        operator = operator.equals("/") ? "div" : operator;

        GetABOpResult abopResult = sdkClient.getABOp(new
        GetABOpRequest().a(x).b(y).op(operator));
        return abopResult.getResult().getOutput();
    }

    public Output getResultWithQueryParameters(String a, String b, String op) {
        GetApiRootResult rootResult = sdkClient.getApiRoot(new
        GetApiRootRequest().a(a).b(b).op(op));
        return rootResult.getResult().getOutput();
    }

    public Output getResultByPostInputBody(Double x, Double y, String o) {
        PostApiRootResult postResult = sdkClient.postApiRoot(
            new PostApiRootRequest().input(new Input().a(x).b(y).op(o)));
        return postResult.getResult().getOutput();
    }

    public static void main( String[] args )
    {
        System.out.println( "Simple calc" );
        // to begin
        App calc = new App();

        // call the SimpleCalc API

```

```
Output res = calc.getResultWithPathParameters("1", "2", "-");
System.out.printf("GET /1/2/-: %s\n", res.getC());

// Use the type query parameter
res = calc.getResultWithQueryParameters("1", "2", "+");
System.out.printf("GET /?a=1&b=2&op=+: %s\n", res.getC());

// Call POST with an Input body.
res = calc.getResultByPostInputBody(1.0, 2.0, "*");
System.out.printf("PUT /\n\n{\"a\":1, \"b\":2,\"op\":\"*\"}\n %s\n",
res.getC());

    }
}
```

Im vorherigen Beispiel dienen die zum Instanzieren des SDK-Clients verwendeten Konfigurationseinstellungen zur Veranschaulichung und sind nicht notwendigerweise als empfohlene bewährte Methode zu verstehen. Darüber hinaus erfolgt das Aufrufen von `sdkClient.shutdown()` optional, besonders dann, wenn Sie eine genaue Kontrolle über das Freisetzen von Ressourcen benötigen.

Wir haben die wichtigsten Muster zum Aufrufen einer API über ein Java SDK gezeigt. Sie können die Anweisungen auch zum Aufrufen anderer API-Methoden verwenden.

Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden

In diesem Abschnitt beschreiben wir die Schritte zur Verwendung eines Android-SDKs, das mit API Gateway für eine REST-API generiert wurde. Bevor Sie fortfahren, müssen Sie die unter [Generieren Sie SDKs für REST APIs in API Gateway](#) beschriebenen Schritte ausführen.

Note

Das generierte SDK ist nicht mit Android 4.4 und früheren Versionen kompatibel. Weitere Informationen finden Sie unter [the section called "Wichtige Hinweise"](#).

So installieren und verwenden Sie ein Android-SDK, das von API Gateway generiert wurde:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.
2. Laden Sie [Apache Maven](#) (vorzugsweise Version 3.x) herunter und installieren Sie das Programm.
3. Laden Sie [JDK 8](#) herunter und installieren Sie die Software.
4. Legen Sie die JAVA_HOME-Umgebungsvariable fest.
5. Führen Sie den Befehl `mvn install` aus, um die kompilierten Artefaktdateien in Ihrem lokalen Maven-Repository zu installieren. Dadurch wird ein Ordner namens `target` erzeugt, der die kompilierte SDK-Bibliothek enthält.
6. Kopieren Sie die SDK-Datei (ihr Name leitet sich von der Artifact-ID und der Artifact-Version ab, die Sie beim Generieren des SDK angegeben haben, z. B. `simple-calcsdk-1.0.0.jar`) aus dem Ordner `target` sowie alle anderen Bibliotheken aus dem Ordner `target/lib` in den Ordner `lib` in Ihrem Projekt.

Bei Verwendung von Android Studio erstellen Sie unterhalb des Client-App-Moduls einen Ordner namens `libs` und kopieren Sie die erforderliche JAR-Datei in diesen Ordner. Überprüfen Sie, ob der Abschnitt mit den Abhängigkeiten in der `gradle`-Datei folgende Informationen enthält.

```
compile fileTree(include: ['*.jar'], dir: 'libs')
compile fileTree(include: ['*.jar'], dir: 'app/libs')
```

Stellen Sie sicher, dass keine doppelten JAR-Dateien deklariert werden.

7. Verwenden Sie die `ApiClientFactory`-Klasse, um das von API Gateway generierte SDK zu initialisieren. Zum Beispiel:

```
ApiClientFactory factory = new ApiClientFactory();

// Create an instance of your SDK. Here, 'SimpleCalcClient.java' is the compiled
// java class for the SDK generated by API Gateway.
final SimpleCalcClient client = factory.build(SimpleCalcClient.class);

// Invoke a method:
// For the 'GET /?a=1&b=2&op=+' method exposed by the API, you can invoke it by
// calling the following SDK method:

Result output = client.rootGet("1", "2", "+");
```

```
//      where the Result class of the SDK corresponds to the Result model of the
//      API.
//

//  For the 'GET /{a}/{b}/{op}' method exposed by the API, you can call the
//  following SDK method to invoke the request,

Result output = client.aBOpGet(a, b, c);

//      where a, b, c can be "1", "2", "add", respectively.

//  For the following API method:
//      POST /
//      host: ...
//      Content-Type: application/json
//
//      { "a": 1, "b": 2, "op": "+" }
//  you can call invoke it by calling the rootPost method of the SDK as follows:
Input body = new Input();
input.a=1;
input.b=2;
input.op="+";
Result output = client.rootPost(body);

//      where the Input class of the SDK corresponds to the Input model of the API.

// Parse the result:
//      If the 'Result' object is { "a": 1, "b": 2, "op": "add", "c":3"}, you
//      retrieve the result 'c') as

String result=output.c;
```

8. Wenn Sie einen Amazon Cognito-Anmeldeinformationsanbieter verwenden möchten, um Aufrufe an Ihre API zu autorisieren, verwenden Sie die `ApiClientFactory`-Klasse, um einen Satz von AWS-Anmeldeinformationen mithilfe des von API Gateway generierten SDKs zu übergeben, wie im folgenden Beispiel gezeigt.

```
// Use CognitoCachingCredentialsProvider to provide AWS credentials
// for the ApiClientFactory
AWSCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
```



```
context,          // activity context
"identityPoolId", // Cognito identity pool id
Regions.US_EAST_1 // region of Cognito identity pool
);

ApiClientFactory factory = new ApiClientFactory()
    .credentialsProvider(credentialsProvider);
```

9. Zum Festlegen eines API-Schlüssels unter Verwendung des von API Gateway generierten SDKs verwenden Sie einen Code ähnlich dem folgenden.

```
ApiClientFactory factory = new ApiClientFactory()
    .apiKey("YOUR_API_KEY");
```

Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API

Das folgende Verfahren zeigt, wie Sie ein von API Gateway generiertes JavaScript SDK verwenden.

 Note

Für diese Anleitungen müssen Sie die Anleitungen unter [Generieren Sie SDKs für REST APIs in API Gateway](#) ausgeführt haben.

 Important

Wenn für Ihre API nur ANY-Methoden definiert wurden, enthält das generierte SDK-Paket keine `apigClient.js`-Datei und Sie müssen die ANY-Methoden selbst definieren.

Um ein von API Gateway generiertes JavaScript SDK für eine REST-API zu installieren, zu initiieren und aufzurufen

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.
2. Aktivieren Sie Cross-Origin Resource Sharing (CORS) für alle Methoden, die das von API Gateway generierte SDK aufrufen werden. Detaillierte Anweisungen finden Sie unter [CORS für REST APIs im API Gateway](#).
3. Bauen Sie in Ihre Webseite Verweise auf die folgenden Scripts ein.

```
<script type="text/javascript" src="lib/axios/dist/axios.standalone.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/hmac-sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/hmac.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/enc-base64.js"></script>
<script type="text/javascript" src="lib/url-template/url-template.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/sigV4Client.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/apiGatewayClient.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/simpleHttpClient.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/utils.js"></script>
<script type="text/javascript" src="apigClient.js"></script>
```

4. Initialisieren Sie in Ihrem Code das von API Gateway generierte SDK, indem Sie Code ähnlich dem folgenden verwenden.

```
var apigClient = apigClientFactory.newClient();
```

Verwenden Sie Code, der dem folgenden ähnelt, um das von API Gateway generierte SDK mit AWS Anmeldeinformationen zu initialisieren. Wenn Sie AWS Anmeldeinformationen verwenden, werden alle Anfragen an die API signiert.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Um einen API-Schlüssel mit dem von API Gateway generierten SDK zu verwenden, übergeben Sie den API-Schlüssel als Parameter an das Factory-Objekt, indem Sie Code ähnlich dem folgenden verwenden. Wenn Sie einen API-Schlüssel verwenden, ist dieser als Teil des `x-api-key`-Headers spezifiziert, und alle Anforderungen an die API werden signiert. Das bedeutet, dass Sie die entsprechenden CORS Accept-Header für jede Anforderung festlegen müssen.

```
var apigClient = apigClientFactory.newClient({
    apiKey: 'API_KEY'
});
```

5. Rufen Sie die API-Methoden in API Gateway auf, indem Sie Code ähnlich dem folgenden verwenden. Jeder Aufruf gibt eine Zusage mit erfolgreichen und fehlgeschlagenen Callbacks zurück.

```
var params = {
    // This is where any modeled request parameters should be added.
    // The key is the parameter name, as it is defined in the API in API Gateway.
    param0: '',
    param1: ''
};

var body = {
    // This is where you define the body of the request,
};

var additionalParams = {
    // If there are any unmodeled query parameters or headers that must be
    // sent with the request, add them here.
    headers: {
        param0: '',
        param1: ''
    },
    queryParams: {
        param0: '',
        param1: ''
    }
};

apigClient.methodName(params, body, additionalParams)
    .then(function(result){
```

```
// Add success callback code here.
}).catch( function(result){
    // Add error callback code here.
});
```

Hier *methodName* wird das aus dem Ressourcenpfad der Methodenanforderung und dem HTTP-Verb erstellt. Für die SimpleCalc API die SDK-Methoden für die API-Methoden von

1. GET `/?a=...&b=...&op=...`
2. POST `/`

`{ "a": ..., "b": ..., "op": ... }`
3. GET `/[a]/[b]/[op]`

Die entsprechenden SDK-Methoden sind:

1. `rootGet(params);` // where `params={"a": ..., "b": ..., "op": ...}` is resolved to the query parameters
2. `rootPost(null, body);` // where `body={"a": ..., "b": ..., "op": ...}`
3. `abOpGet(params);` // where `params={"a": ..., "b": ..., "op": ...}` is resolved to the path parameters

Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden

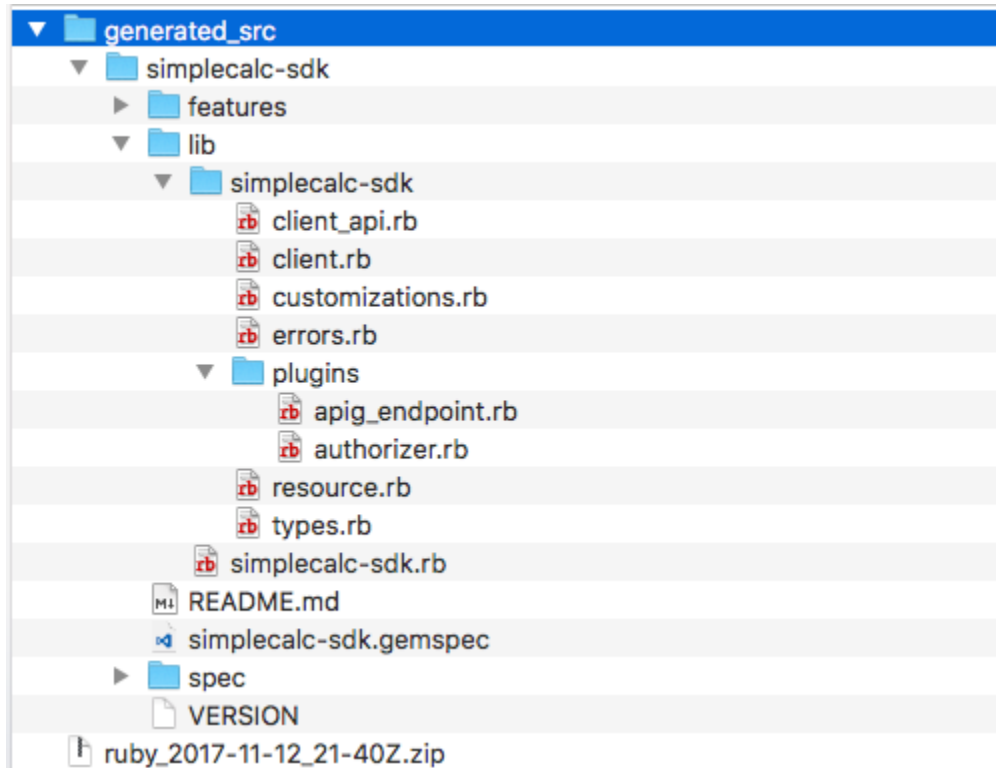
Im folgenden Verfahren wird die Verwendung eines Ruby SDK gezeigt, das mit API Gateway generiert wurde.

Note

Für diese Anleitungen müssen Sie die Anleitungen unter [Generieren Sie SDKs für REST APIs in API Gateway](#) ausgeführt haben.

So installieren und instanziiieren Sie ein von API Gateway generiertes Ruby-SDK für eine REST-API und rufen es auf:

1. Entpacken Sie die heruntergeladene Ruby-SDK-Datei. Die generierte SDK-Quelle wird wie folgt angezeigt.



- Erstellen Sie mit den folgenden Shell-Befehlen einen Ruby-Gem aus der generierten SDK-Quelle in einem Terminalfenster:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

Anschließend ist `simplecalc-sdk-1.0.0.gem` verfügbar.

- Installieren Sie das Gem:

```
gem install simplecalc-sdk-1.0.0.gem
```

- Erstellen Sie eine Client-Anwendung. Instanziiert und initialisiert den Ruby-SDK-Client in der App:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
  http_wire_trace: true,
  retry_limit: 5,
```

```
    http_read_timeout: 50  
  )
```

Wenn die API über eine Autorisierung des konfigurierten AWS_IAM Typs verfügt, können Sie die AWS Anmeldeinformationen des Aufrufers angeben, indem Sie `accessKey` und `secretKey` während der Initialisierung angeben:

```
require 'pet-sdk'  
client = Pet::Client.new(  
  http_wire_trace: true,  
  retry_limit: 5,  
  http_read_timeout: 50,  
  access_key_id: 'ACCESS_KEY',  
  secret_access_key: 'SECRET_KEY'  
)
```

5. Führen Sie API-Aufrufe über das SDK in der App durch.

Tip

Wenn Sie nicht mit den Konventionen des SDK-Methodenaufrufs vertraut sind, können Sie die `client.rb`-Datei im generierten lib-SDK-Ordner überprüfen. Der Ordner enthält die Dokumentation der einzelnen unterstützten API-Methodenaufrufe.

So erkennen Sie unterstützte Operationen:

```
# to show supported operations:  
puts client.operation_names
```

Dies führt entsprechend der API-Methoden GET `/?a={.}&b={.}&op={.}`, GET `/a/b/op` und POST `/` jeweils folgender Anzeige sowie zu einer Nutzlast vom `{a:"...", b:"...", op:"..."}`-Format:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Für den Aufruf der GET `/?a=1&b=2&op=+`-API-Methode rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Für den Aufruf der POST `/-API-Methode` mit einer Nutzlast von `{a: "1", b: "2", "op": "+"}` rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Für den Aufruf der GET `/1/2/+API-Methode` rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Die erfolgreichen SDK-Methodenaufrufe geben die folgende Antwort zurück:

```
resp : {
  result: {
    input: {
      a: 1,
      b: 2,
      op: "+"
    },
    output: {
      c: 3
    }
  }
}
```

Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden

In diesem Tutorial erfahren Sie, wie Sie ein iOS SDK, das von API Gateway für eine REST-API in einer Objective-C- oder Swift-App generiert wurde, zum Aufrufen der zugrunde liegenden API verwenden. Wir werden die [SimpleCalc API](#) als Beispiel verwenden, um die folgenden Themen zu veranschaulichen:

- So installieren Sie die erforderlichen AWS Mobile SDK-Komponenten in Ihrem Xcode-Projekt
- So erstellen Sie das API-Clientobjekt vor dem Aufruf der API-Methoden

- So rufen Sie die API-Methoden mit den entsprechenden SDK-Methoden auf dem API-Clientobjekt auf
- So generieren Sie die Methodeneingabe und analysieren das Ergebnis mit den entsprechenden SDK-Modellklassen

Themen

- [Verwenden des generierten iOS-SDK \(Objective-C\) zum Aufrufen der API](#)
- [Verwenden eines generierten iOS-SDK \(Swift\) zum Aufrufen der API](#)

Verwenden des generierten iOS-SDK (Objective-C) zum Aufrufen der API

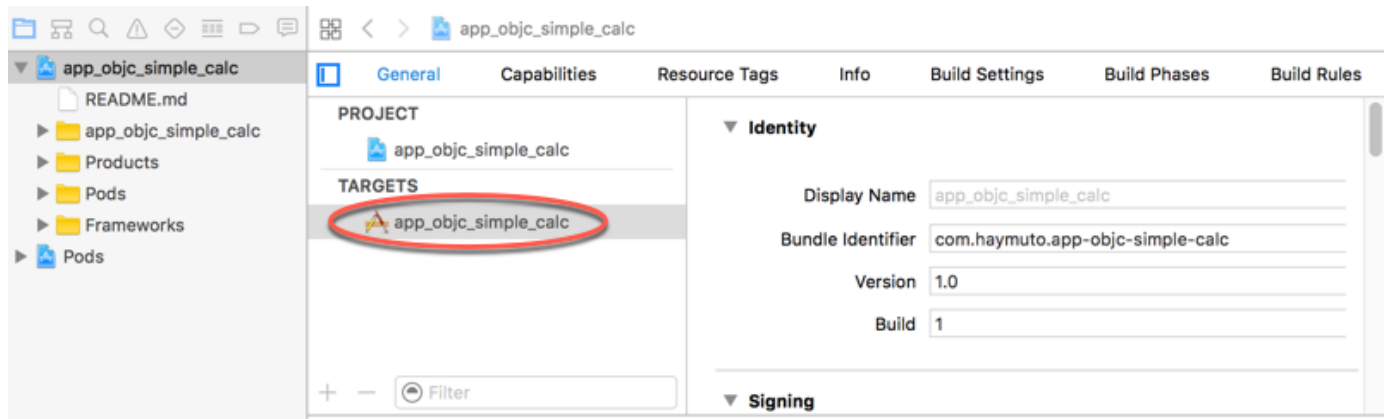
Bevor Sie mit folgendem Verfahren beginnen, müssen Sie die Schritte in [Generieren Sie SDKs für REST APIs in API Gateway](#) für iOS in Objective-C durchführen und die ZIP-Datei des generierten SDK herunterladen.

Installieren Sie das AWS mobile SDK und ein von API Gateway generiertes iOS-SDK in einem Objective-C-Projekt

Im folgenden Verfahren wird beschrieben, wie Sie das SDK installieren.

So installieren und verwenden Sie ein iOS-SDK, das von API Gateway generiert wurde, in Objective-C:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben. Wenn Sie die [SimpleCalc API](#) verwenden, möchten Sie den entpackten SDK-Ordner möglicherweise in etwas wie umbenennen. `sdk_objc_simple_calc` In diesem SDK-Ordner sind eine `README.md`-Datei und eine `Podfile`-Datei enthalten. Die `README.md`-Datei enthält die Anweisungen für die Installation und Verwendung des SDK. Dieses Tutorial bietet Details zu diesen Anweisungen. Die Installation nutzt [CocoaPods](#) den Import der erforderlichen API Gateway Gateway-Bibliotheken und anderer abhängiger AWS Mobile SDK-Komponenten. Sie müssen das aktualisieren `Podfile`, um es SDKs in das Xcode-Projekt Ihrer App zu importieren. Der nicht archivierte SDK-Ordner enthält auch einen `generated-src`-Ordner mit dem Quellcode des generierten SDK Ihrer API.
2. Starten Sie Xcode und erstellen Sie ein neues iOS Objective-C-Projekt. Notieren Sie das Projektziel. Sie müssen die entsprechende Einstellung in der `Podfile` vornehmen.



3. Gehen Sie wie folgt vor, um das mithilfe CocoaPods von AWS Mobile SDK for iOS in das Xcode-Projekt zu importieren:

a. Führen Sie zur Installation CocoaPods den folgenden Befehl in einem Terminalfenster aus:

```
sudo gem install cocoapods
pod setup
```

b. Kopieren Sie die Podfile-Datei vom extrahierten SDK-Ordner in dasselbe Verzeichnis, das Ihre Xcode Projektdatei enthält. Ersetzen Sie den folgenden Block:

```
target '<YourXcodeTarget>' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

mit dem Zielnamen Ihres Projekts:

```
target 'app_objc_simple_calc' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Wenn Ihr Xcode-Projekt bereits eine Datei mit dem Namen Podfile enthält, fügen Sie die folgende Codezeile hinzu:

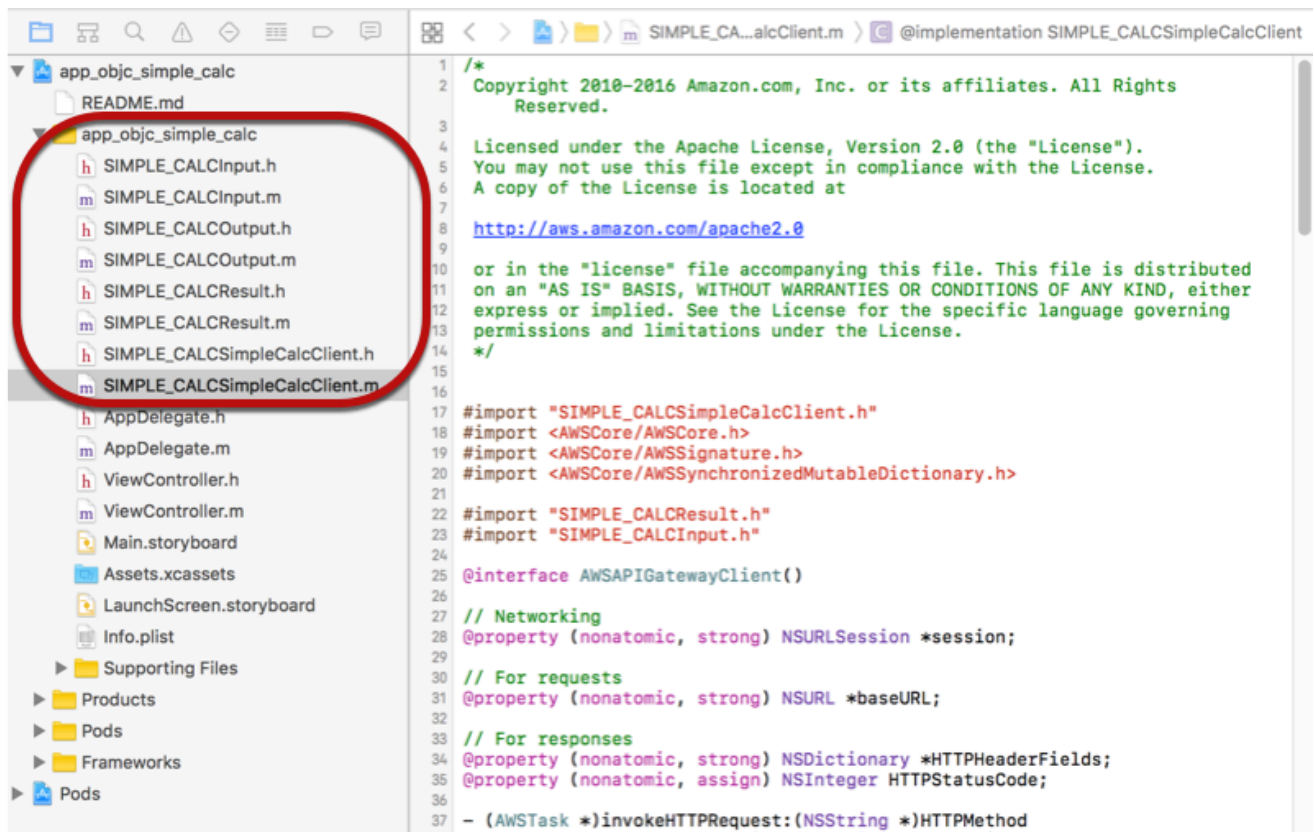
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

c. Öffnen Sie ein Terminalfenster und führen Sie den folgenden Befehl aus:

```
pod install
```

Dadurch werden die API Gateway Gateway-Komponente und andere abhängige AWS Mobile SDK-Komponenten installiert.

- d. Schließen Sie das Xcode-Projekt und öffnen Sie anschließend die `.xcworkspace`-Datei, um Xcode neu zu starten.
- e. Fügen Sie alle `.h`- und `.m`-Dateien aus dem extrahierten SDK-Verzeichnis `generated-src` Ihrem Xcode-Projekt hinzu.



Um AWS Mobile SDK for iOS Objective-C in Ihr Projekt zu importieren, indem Sie das AWS Mobile SDK explizit herunterladen oder [Carthage](#) verwenden, folgen Sie den Anweisungen in der Datei `README.md`. Achten Sie darauf, nur eine dieser Optionen zu verwenden, um das Mobile SDK zu importieren. AWS

In einem Objective-C-Projekt API-Methoden unter Verwendung des von API Gateway generierten iOS-SDKs aufrufen

Wenn Sie das SDK mit dem Präfix `SIMPLE_CALC` für diese [SimpleCalc API](#) mit zwei Modellen für Eingabe (Input) und Ausgabe (Result) der Methoden generiert haben, wird im SDK die

resultierende API-Clientklasse `SIMPLE_CALCSimpleCalcClient` und die entsprechenden Datenklassen sind jeweils `SIMPLE_CALCInput` und `SIMPLE_CALCResult`. Die API-Anfragen und Antworten sind den SDK-Methoden wie folgt zugeordnet:

- Die API-Anfrage

```
GET /?a=...&b=...&op=...
```

wird die SDK-Methode

```
(AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
```

Die Eigenschaft `AWSTask.result` ist vom Typ `SIMPLE_CALCResult`, wenn das Modell `Result` zur Methodenantwort hinzugefügt wurde. Andernfalls ist die Eigenschaft vom Typ `NSDictionary`.

- Diese API-Anfrage

```
POST /

{
  "a": "Number",
  "b": "Number",
  "op": "String"
}
```

wird die SDK-Methode

```
(AWSTask *)rootPost:(SIMPLE_CALCInput *)body
```

- Die API-Anfrage

```
GET /{a}/{b}/{op}
```

wird die SDK-Methode

```
(AWSTask *)aB0pGet:(NSString *)a b:(NSString *)b op:(NSString *)op
```

Im folgenden Verfahren wird beschrieben, wie die API-Methoden im Objective-C-App-Quellcode aufgerufen werden, z. B. als Teil des `viewDidLoad`-Delegaten in einer `ViewController.m`-Datei.

So rufen Sie die API über das von API Gateway generierte iOS-SDK auf:

1. Importieren Sie die Header-Datei der API-Client-Klasse, um die API-Client-Klasse in der App aufrufbar zu machen:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

Durch die Anweisung `#import` werden auch `SIMPLE_CALCInput.h` und `SIMPLE_CALCResult.h` für die beiden Modellklassen importiert.

2. Instanzieren Sie die API-Client-Klasse:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
    defaultClient];
```

Um Amazon Cognito mit der API zu verwenden, legen Sie, wie im Folgenden gezeigt, Sie die `defaultServiceConfiguration`-Eigenschaft auf das `StandardAWSServiceManager`-Objekt fest, bevor Sie die `defaultClient`-Methode aufrufen, um das API-Client-Objekt zu erstellen (im vorhergehenden Beispiel gezeigt):

```
AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
    initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
    configuration;
```

3. Rufen Sie die GET `/?a=1&b=2&op=+-`-Methode auf, um `1+2` durchzuführen:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
    * _Nonnull task) {
    _textField1.text = [self handleApiResponse:task];
    return nil;
}];
```

in der die Hilfsfunktion `handleApiResponse:task` das Ergebnis als eine Zeichenfolge formatiert, die in einem Textfeld (`_textField1`) angezeigt werden soll.

```
- (NSString *)handleApiResponse:(AWSTask *)task {
    if (task.error != nil) {
```

```

        return [NSString stringWithFormat: @"Error: %@", task.error.description];
    } else if (task.result != nil && [task.result isKindOfClass:[SIMPLE_CALCResult
class]]) {
        return [NSString stringWithFormat:@"%d %d %d = %d\n",task.result.input.a,
task.result.input.op, task.result.input.b, task.result.output.c];
    }
    return nil;
}

```

Die sich ergebende Anzeige ist $1 + 2 = 3$.

4. Rufen Sie den POST / mit einer Nutzlast aus, um 1-2 durchzuführen:

```

SIMPLE_CALCInput *input = [[SIMPLE_CALCInput alloc] init];
input.a = [NSNumber numberWithInt:1];
input.b = [NSNumber numberWithInt:2];
input.op = @"-";
[[apiInstance rootPost:input] continueWithBlock:^(id _Nullable(AWSTask *
_Nonnull task) {
    _textField2.text = [self handleApiResponse:task];
    return nil;
}]);

```

Die sich ergebende Anzeige ist $1 - 2 = -1$.

5. Rufen Sie GET /{a}/{b}/{op} auf, um 1/2 durchzuführen:

```

[[apiInstance aB0pGet:@"1" b:@"2" op:@"div"] continueWithBlock:^(id
_Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}]);

```

Die sich ergebende Anzeige ist $1 \text{ div } 2 = 0.5$. Hier wird div anstelle von / verwendet, da die [einfache Lambda-Funktion](#) im Backend keine URL-kodierten Pfadvariablen verarbeitet.

Verwenden eines generierten iOS-SDK (Swift) zum Aufrufen der API

Bevor Sie mit folgendem Verfahren beginnen, müssen Sie die Schritte in [Generieren Sie SDKs für REST APIs in API Gateway](#) für iOS in Swift durchführen und die ZIP-Datei des generierten SDK herunterladen.

Themen

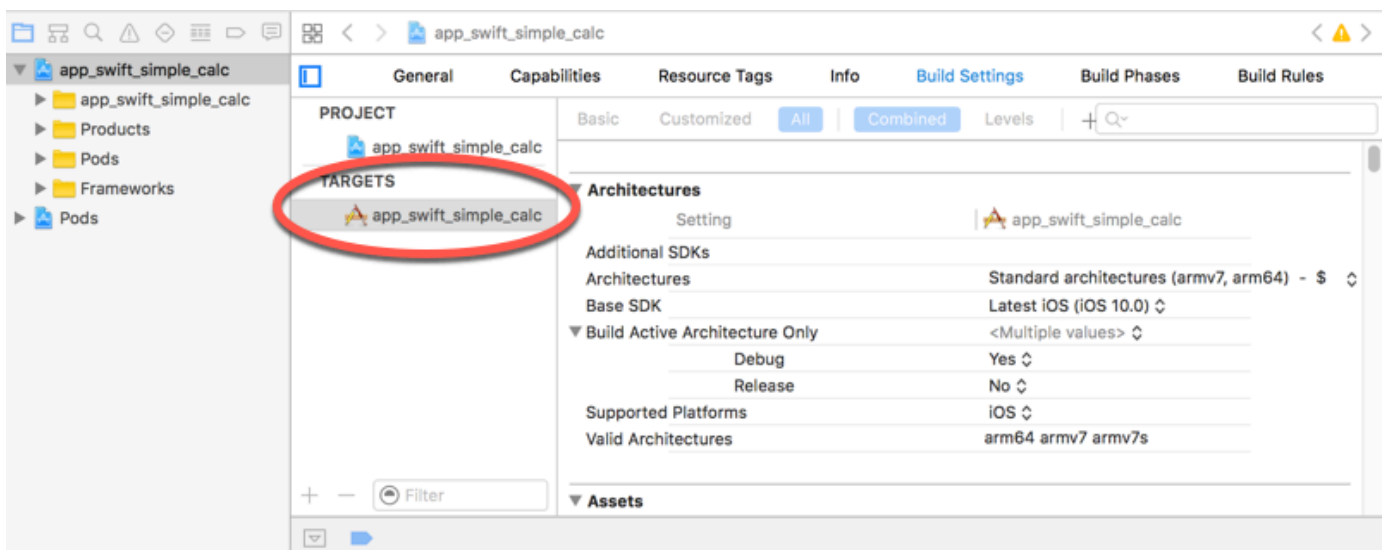
- [Installieren Sie das AWS mobile SDK und das vom API Gateway generierte SDK in einem Swift-Projekt](#)
- [API-Methoden über das von API Gateway generierte iOS-SDK in einem Swift-Projekt aufrufen](#)

Installieren Sie das AWS mobile SDK und das vom API Gateway generierte SDK in einem Swift-Projekt

Im folgenden Verfahren wird beschrieben, wie Sie das SDK installieren.

So installieren und verwenden Sie von API Gateway generiertes iOS-SDK in Swift:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben. Wenn Sie die [SimpleCalc API](#) verwenden, möchten Sie den entpackten SDK-Ordner möglicherweise in etwas Ähnliches umbenennen. **sdk_swift_simple_calc**
In diesem SDK-Ordner sind eine `README.md`-Datei und eine `Podfile`-Datei enthalten. Die `README.md`-Datei enthält die Anweisungen für die Installation und Verwendung des SDK. Dieses Tutorial bietet Details zu diesen Anweisungen. Die Installation nutzt [CocoaPods](#) den Import der erforderlichen AWS Mobile SDK-Komponenten. Sie müssen das aktualisieren `Podfile`, um es SDKs in das Xcode-Projekt Ihrer Swift-App zu importieren. Der nicht archivierte SDK-Ordner enthält auch einen `generated-src`-Ordner mit dem Quellcode des generierten SDK Ihrer API.
2. Starten Sie Xcode und erstellen Sie ein neues iOS Swift-Projekt. Notieren Sie das Projektziel. Sie müssen die entsprechende Einstellung in der `Podfile` vornehmen.



3. Gehen Sie wie folgt vor, um die erforderlichen AWS Mobile SDK-Komponenten mithilfe CocoaPods von in das Xcode-Projekt zu importieren:
 - a. Wenn es nicht installiert ist, installieren Sie es, CocoaPods indem Sie den folgenden Befehl in einem Terminalfenster ausführen:
- b. Kopieren Sie die Podfile-Datei vom extrahierten SDK-Ordner in dasselbe Verzeichnis, das Ihre Xcode Projektdatei enthält. Ersetzen Sie den folgenden Block:

```
sudo gem install cocoapods
pod setup
```

```
target '<YourXcodeTarget>' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

mit dem Zielnamen Ihres Projekts wie dargestellt:

```
target 'app_swift_simple_calc' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Wenn Ihr Xcode-Projekt bereits eine Podfile mit dem korrekten Ziel enthält, können Sie einfach die folgende Codezeile zur `do ... end`-Schleife hinzufügen:

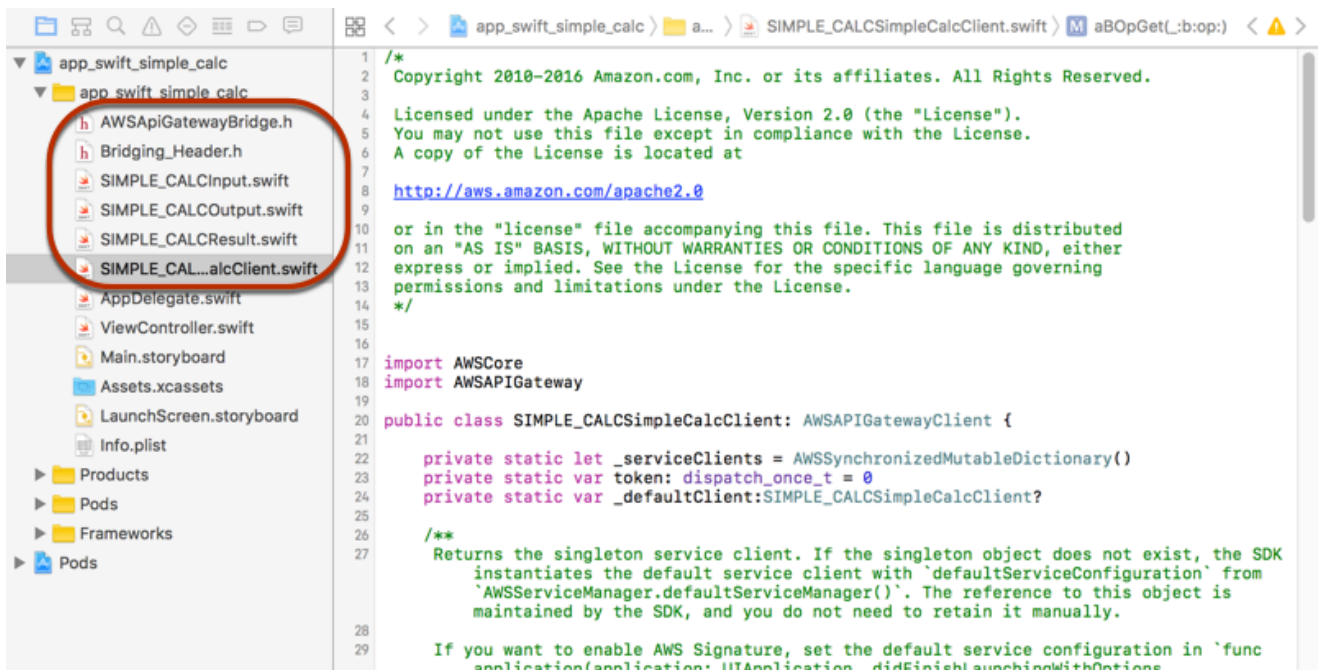
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

- c. Öffnen Sie ein Terminalfenster und führen Sie den folgenden Befehl im App-Verzeichnis aus:

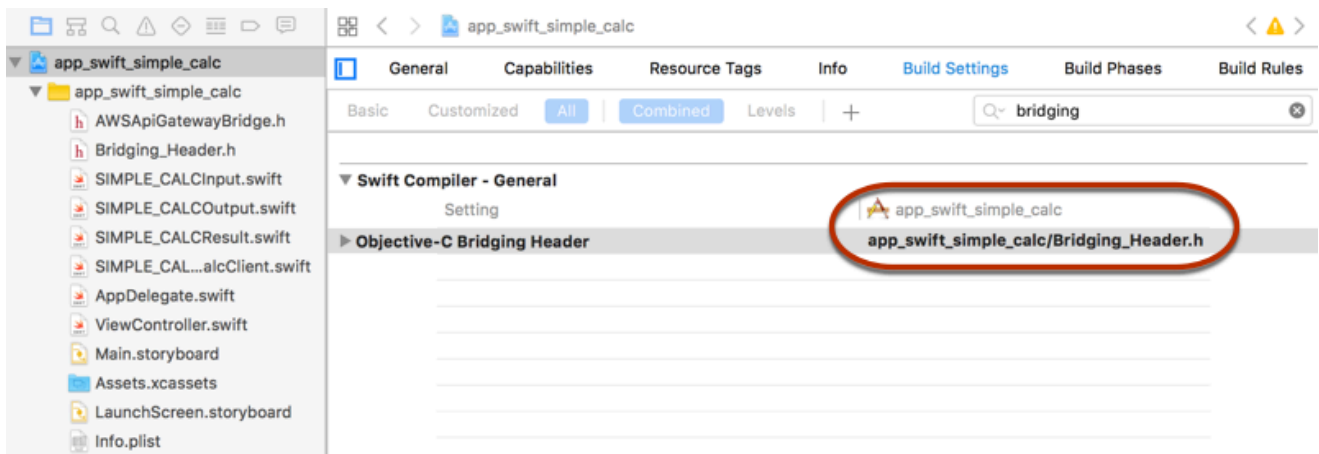
```
pod install
```

Dadurch werden die API Gateway Gateway-Komponente und alle abhängigen AWS Mobile SDK-Komponenten im Projekt der App installiert.

- d. Schließen Sie das Xcode-Projekt und öffnen Sie anschließend die `*.xcworkspace`-Datei, um Xcode neu zu starten.
- e. Fügen Sie alle SDK-Header-Dateien (`.h`) und Swift-Quelldateien (`.swift`) aus dem extrahierten `generated-src`-Verzeichnis zu Ihrem Xcode-Projekt hinzu.



- f. Um das Aufrufen der Objective-C-Bibliotheken des AWS Mobile SDK von Ihrem Swift-Code-Projekt aus zu ermöglichen, legen Sie den Bridging_Header.h Dateipfad in der Eigenschaft Objective-C Bridging Header unter der Einstellung Swift Compiler — General Ihrer Xcode-Projektkonfiguration fest:

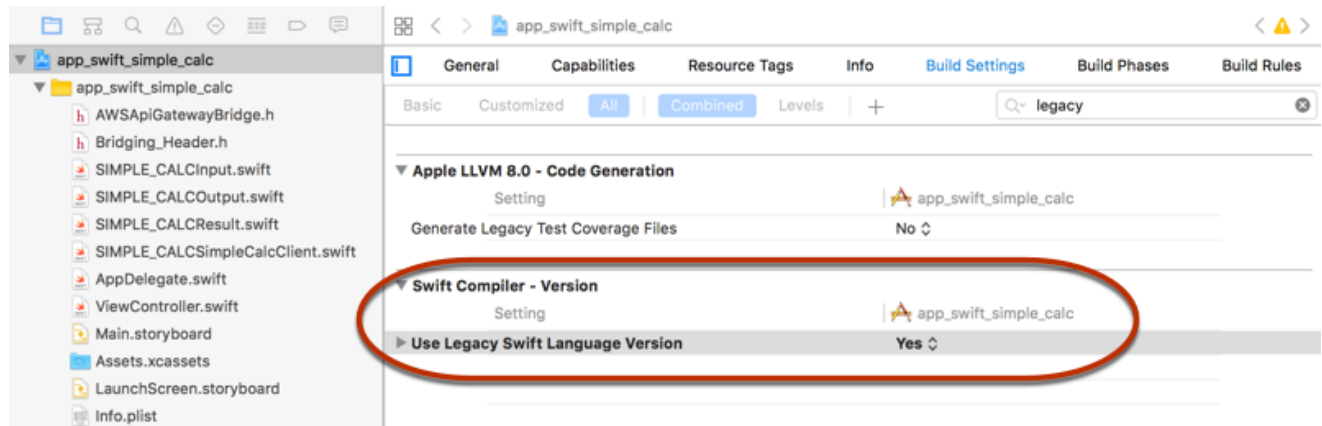


Tip

Sie können im Suchfeld von Xcode **bridging** eingeben, um die Eigenschaft Objective-C Bridging Header zu suchen.

- g. Erstellen Sie das Xcode-Projekt, um zu überprüfen, dass es ordnungsgemäß konfiguriert wurde, bevor Sie fortfahren. Wenn Ihr Xcode eine neuere Version von Swift verwendet als

die, die für das AWS Mobile SDK unterstützt wird, werden Swift-Compilerfehler angezeigt. Setzen Sie in diesem Fall die Eigenschaft Use Legacy Swift Language Version auf Yes in der Einstellung Swift Compiler- Version:



Um das AWS Mobile SDK for iOS in Swift in Ihr Projekt zu importieren, indem Sie das AWS Mobile SDK explizit herunterladen oder [Carthage](#) verwenden, folgen Sie den Anweisungen in der README.md Datei, die dem SDK-Paket beiliegt. Achten Sie darauf, nur eine dieser Optionen zu verwenden, um das AWS Mobile SDK zu importieren.

API-Methoden über das von API Gateway generierte iOS-SDK in einem Swift-Projekt aufrufen

Wenn Sie das SDK mit dem Präfix SIMPLE_CALC für diese [SimpleCalc API](#) mit zwei Modellen zur Beschreibung der Eingabe (Input) und Ausgabe (Result) der Anfragen und Antworten der API generiert haben, wird im SDK die resultierende API-Clientklasse SIMPLE_CALCSimpleCalcClient und die entsprechenden Datenklassen sind jeweils SIMPLE_CALCInput und SIMPLE_CALCResult. Die API-Anfragen und-Antworten sind den SDK-Methoden wie folgt zugeordnet:

- Die API-Anfrage

```
GET /?a=...&b=...&op=...
```

wird die SDK-Methode

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

Die Eigenschaft `AWSTask.result` ist vom Typ `SIMPLE_CALCResult`, wenn das Modell `Result` zur Methodenantwort hinzugefügt wurde. Andernfalls ist sie vom Typ `NSDictionary`.

- Diese API-Anfrage

```
POST /

{
  "a": "Number",
  "b": "Number",
  "op": "String"
}
```

wird die SDK-Methode

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- Die API-Anfrage

```
GET /{a}/{b}/{op}
```

wird die SDK-Methode

```
public func aB0pGet(a: String, b: String, op: String) -> AWSTask
```

Im folgenden Verfahren wird beschrieben, wie die API-Methoden im Swift-App-Quellcode aufgerufen werden, z. B. als Teil des `viewDidLoad()`-Delegaten in einer `ViewController.m`-Datei.

So rufen Sie die API über das von API Gateway generierte iOS-SDK auf:

1. Instanzieren Sie die API-Client-Klasse:

```
let client = SIMPLE_CALCSimpleCalcClient.default()
```

Um Amazon Cognito mit der API zu verwenden, legen Sie eine AWS Standard-Servicekonfiguration fest (siehe unten), bevor Sie die `default` Methode aufrufen (siehe oben):

```
let credentialsProvider =
  AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
    "my_pool_id")
let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
  credentialsProvider: credentialsProvider)
```

```
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
    configuration
```

2. Rufen Sie die GET `/?a=1&b=2&op=+-` Methode auf, um 1+2 durchzuführen:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
in
    self.showResult(task)
    return nil
}
```

in der die Hilfsfunktion `self.showResult(task)` das Ergebnis oder den Fehler auf die Konsole druckt, z. B.:

```
func showResult(task: AWSTask) {
    if let error = task.error {
        print("Error: \(error)")
    } else if let result = task.result {
        if result is SIMPLE_CALCResult {
            let res = result as! SIMPLE_CALCResult
            print(String(format:"%@ %@ %@ = %@", res.input!.a!, res.input!.op!,
res.input!.b!, res.output!.c!))
        } else if result is NSDictionary {
            let res = result as! NSDictionary
            print("NSDictionary: \(res)")
        }
    }
}
```

In einer Produktion-App können Sie das Ergebnis oder den Fehler in einem Textfeld anzeigen. Die sich ergebende Anzeige ist `1 + 2 = 3`.

3. Rufen Sie den POST `/` mit einer Nutzlast aus, um 1-2 durchzuführen:

```
let body = SIMPLE_CALCInput()
body.a=1
body.b=2
body.op="-"
client.rootPost(body).continueWithBlock {(task: AWSTask) -> AnyObject? in
    self.showResult(task)
    return nil
}
```

Die resultierende Anzeige ist $1 - 2 = -1$.

4. Rufen Sie GET `/a/b/op` auf, um $1/2$ durchzuführen:

```
client.aB0pGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
  AnyObject? in
    self.showResult(task)
    return nil
}
```

Die sich ergebende Anzeige ist $1 \text{ div } 2 = 0.5$. Hier wird `div` anstelle von `/` verwendet, da die [einfache Lambda-Funktion](#) im Backend keine URL-kodierten Pfadvariablen verarbeitet.

REST-APIs mit OpenAPI in API Gateway erstellen

Sie können API Gateway verwenden, um eine REST-API aus einer externen Definitionsdatei in API Gateway zu importieren. Gegenwärtig unterstützt API Gateway die Definitionsdateien [OpenAPI v2.0](#) und [OpenAPI v3.0](#), wobei Ausnahmen in [Amazon API Gateway — wichtige Hinweise für REST APIs](#) aufgeführt werden. Sie können eine API aktualisieren, indem Sie diese mit einer neuen Definition überschreiben, oder Sie können eine Definition mit einer vorhandenen API zusammenführen. Die Optionen in der Anforderungs-URL geben Sie mithilfe eines mode-Abfrageparameters an.

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Themen

- [Edge-optimierte API in API Gateway importieren](#)
- [Regionale API in API Gateway importieren](#)
- [Importieren einer OpenAPI-Datei zum Aktualisieren einer vorhandenen API-Definition](#)
- [Legen Sie die OpenAPI-BasePath-Eigenschaft fest](#)
- [AWS Variablen für OpenAPI-Import](#)
- [Fehler und Warnungen beim Import Ihrer API in API Gateway](#)
- [REST-API von API Gateway importieren](#)

Edge-optimierte API in API Gateway importieren

Geben Sie neben der OpenAPI-Datei für den Importvorgang den Endpunkttyp EDGE als zusätzliche Eingabe an, um eine API-OpenAPI-Datei zum Erstellen einer neuen Edge-optimierten API zu importieren. Sie können dies mit der API Gateway Gateway-Konsole oder einem AWS SDK tun. AWS CLI

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Themen

- [Edge-optimierte API über die API Gateway-Konsole importieren](#)
- [Importieren Sie eine Edge-optimierte API mit dem AWS CLI](#)

Edge-optimierte API über die API Gateway-Konsole importieren

Gehen Sie wie folgt vor, um eine Edge-optimierte API mit der API-Gateway-Konsole zu importieren:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) aus.
3. Unter REST-API, wählen Sie Import (Importieren) aus.
4. Kopieren Sie die OpenAPI-Definition einer API und fügen Sie sie in den Code-Editor ein, oder klicken Sie auf Choose file (Datei auswählen), um eine OpenAPI-Datei von einem lokalen Laufwerk zu laden.
5. Wählen Sie für API-Endpunkt-Typ die Option Edge-optimiert aus.
6. Wählen Sie Create API (API erstellen) aus, um die OpenAPI-Definitionen zu importieren.

Importieren Sie eine Edge-optimierte API mit dem AWS CLI

Der folgende [import-rest-api](#)Befehl importiert eine API aus einer OpenAPI-Definitionsdatei, um eine neue Edge-optimierte API zu erstellen:

```
aws apigateway import-rest-api \  
  --fail-on-warnings \  
  --body 'file://path/to/API_OpenAPI_template.json'
```

Oder geben Sie für den Abfragezeichenfolgenparameter `endpointConfigurationTypes` ausdrücklich `EDGE` an:

```
aws apigateway import-rest-api \  
  --parameters endpointConfigurationTypes=EDGE \  
  --fail-on-warnings \  
  --body 'file://path/to/API_OpenAPI_template.json'
```

Regionale API in API Gateway importieren

Beim Importieren einer API können Sie die regionale Endpunktconfiguration für die API auswählen. Sie können die API Gateway Gateway-Konsole AWS CLI, das oder ein AWS SDK verwenden.

Die API-Endpunktconfiguration ist beim Exportieren einer API nicht in den exportierten API-Definitionen enthalten.

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Themen

- [Regionale API über die API Gateway-Konsole importieren](#)
- [Importieren Sie eine regionale API mit dem AWS CLI](#)

Regionale API über die API Gateway-Konsole importieren

Gehen Sie wie folgt vor, um eine API eines regionalen Endpunkts über die API Gateway-Konsole zu importieren:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie **Create API (API erstellen)** aus.
3. Unter **REST-API**, wählen Sie **Import (Importieren)** aus.
4. Kopieren Sie die OpenAPI-Definition einer API und fügen Sie sie in den Code-Editor ein, oder klicken Sie auf **Choose file (Datei auswählen)**, um eine OpenAPI-Datei von einem lokalen Laufwerk zu laden.
5. Wählen Sie für API-Endpunkttyp die Option **Regional** aus.

6. Wählen Sie **Create API (API erstellen)** aus, um die OpenAPI-Definitionen zu importieren.

Importieren Sie eine regionale API mit dem AWS CLI

Der folgende [import-rest-api](#) Befehl importiert eine OpenAPI-Definitionsdatei und setzt den Endpunkttyp auf Regional:

```
aws apigateway import-rest-api \  
  --parameters endpointConfigurationTypes=REGIONAL \  
  --fail-on-warnings \  
  --body 'file:///path/to/API_OpenAPI_template.json'
```

Importieren einer OpenAPI-Datei zum Aktualisieren einer vorhandenen API-Definition

Sie können nur API-Definitionen importieren, um eine vorhandene API zu aktualisieren, ohne die Endpunkt-Konfiguration sowie Phasen und Phasen-Variablen oder Referenzen zu API-Schlüsseln zu verändern.

Der Importvorgang zum Aktualisieren kann auf zwei Arten erfolgen: durch Zusammenführen oder Überschreiben.

Wenn eine API (A) mit einer anderen (B) zusammengeführt wird, behält die resultierende API die Definitionen von A und B bei, wenn die beiden APIs keine widersprüchlichen Definitionen aufweisen. Wenn Konflikte auftreten, überschreiben die Methodendefinitionen der zusammenführenden API (A) die entsprechenden Methodendefinitionen der zusammengeführten API (B). Beispiel: B hat die folgenden Methoden deklariert, um die Antworten 200 und 206 zurückzugeben:

```
GET /a  
POST /a
```

und A deklariert die folgende Methode, um die Antworten 200 und 400 zurückzugeben:

```
GET /a
```

Wenn A mit B zusammengeführt wird, ergibt die resultierende API die folgenden Methoden:

```
GET /a
```

die 200- und 400-Antworten ausgibt, und

```
POST /a
```

die 200- und 206-Antworten ausgibt.

Die Zusammenführung einer API empfiehlt sich, wenn Sie die externen API-Definitionen in mehrere kleinere Teile zerlegt haben und nur die Änderungen eines dieser Teile anwenden möchten. Dies kann z. B. der Fall sein, wenn für die verschiedenen Teile einer API mehrere Teams zuständig sind, die in unterschiedlichem Tempo Änderungen vornehmen möchten. In diesem Modus bleiben Elemente einer vorhandenen API, die in der importierten Definition nicht speziell definiert wurden, als eigenständige Elemente erhalten.

Wenn eine API (A) eine andere API (B) überschreibt, übernimmt die resultierende API die Definitionen der überschreibenden API (A). Das Überschreiben einer API empfiehlt sich, wenn eine externe API-Definition die vollständige Definition einer API enthält. In diesem Modus werden Elemente einer vorhandenen API, die in der importierten Definition nicht speziell definiert wurden, gelöscht.

Important

Wenn Sie `mode=overwrite` verwenden, wird die gesamte API-Definition ersetzt. Alle API-Konfigurationen, die nicht explizit im Importtext enthalten sind, werden auf ihre Standardwerte zurückgesetzt. Dazu gehören Einstellungen wie Sicherheitsrichtlinien, Autorisierungskonfigurationen und andere API-level Eigenschaften.

Um bestehende Konfigurationen beizubehalten, verwenden Sie `mode=merge`, wodurch bestehende Konfigurationen beibehalten werden, die nicht im Importtext angegeben sind.

Wenn Sie `mode=overwrite` verwenden, stellen Sie sicher, dass Ihre OpenAPI-Definition alle gewünschten API-Konfigurationen enthält, z. B. API-Gateway-Erweiterungen.

Wenn Sie eine API zusammenführen möchten, übermitteln Sie eine PUT-Anforderung an `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. Der Wert `restapi_id` des Parameters "path" gibt die API an, mit der die bereitgestellte API-Definition zusammengeführt wird.

Das folgende Codefragment zeigt ein Beispiel für die PUT-Anfrage zum Zusammenführen einer OpenAPI-Definition in API – als Nutzlast – mit der angegebenen JSON, die sich bereits in API Gateway befindet.


```
PUT /restapis/<restapi_id>?mode=merge
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...
```

[An OpenAPI API definition in JSON](#)

Bei einer Aktualisierung mittels Zusammenführung werden zwei vollständige API-Definitionen miteinander verbunden. Für geringfügige und inkrementelle Änderungen können Sie die Operation zur [Ressourcenaktualisierung](#) verwenden.

Wenn Sie eine API überschreiben möchten, übermitteln Sie eine PUT-Anforderung an `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. Der Wert `restapi_id` des Parameters "path" gibt die API an, die durch die bereitgestellten API-Definitionen überschrieben wird.

Der folgende Codeausschnitt zeigt ein Beispiel für eine Überschreibanforderung mit der Nutzlast einer Definition: JSON-formatted OpenAPI

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...
```

[An OpenAPI API definition in JSON](#)

Wenn der Abfrageparameter `mode` fehlt, wird von einer Zusammenführung ausgegangen.

Note

Die PUT-Operationen sind idempotent, aber nicht atomar. Das heißt, wenn während der Verarbeitung ein Systemfehler auftritt, kann die API letztlich einen fehlerhaften Zustand annehmen. Bei erfolgreicher wiederholter Ausführung der Operation nimmt die API jedoch den endgültigen Zustand an, den sie auch bei erfolgreicher Ausführung der ersten Operation erreicht hätte.

Legen Sie die **OpenAPI-BasePath-Eigenschaft** fest

In [OpenAPI 2.0](#) können Sie mit der `basePath`-Eigenschaft einen oder mehrere Pfadbestandteile bereitstellen, die jedem in den `paths`-Eigenschaften definierten Pfad vorangestellt werden. In API Gateway gibt es mehrere Möglichkeiten, den Pfad einer Ressource auszudrücken daher bietet auch die Funktion zum Importieren von APIs drei Optionen für die Interpretation der `basePath`-Eigenschaft während eines Importvorgangs: Ignorieren, Voranstellen und Splitten.

In [OpenAPI 3.0](#) ist `basePath` keine Top-Level-Eigenschaft mehr. Stattdessen verwendet API Gateway eine [Servervariable](#) als Konvention. Die Funktion zum Importieren von APIs bietet die gleichen Optionen für die Interpretation der Basispfades während des Imports. Der Basispfad wird wie folgt identifiziert:

- Wenn die API keine `basePath`-Variablen aufweist, überprüft die Funktion zum Importieren von APIs, ob die `server.url`-Zeichenfolge einen Pfad außer `"/"` enthält. Wenn dies der Fall ist, wird dieser Pfad als Basispfad verwendet.
- Wenn die API nur eine `basePath`-Variable enthält, verwendet die Funktion zum Importieren von APIs diese als Basispfad, auch wenn sie nicht in `server.url` referenziert wird.
- Wenn die API mehrere `basePath`-Variablen enthält, verwendet die Funktion zum Importieren von APIs nur den ersten Eintrag als Basispfad.

Ignore

Wenn in der OpenAPI-Datei der `basePath`-Wert `/a/b/c` lautet und die `paths`-Eigenschaft `/e` und `/f` enthält, führt die folgende POST- oder PUT-Anforderung:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/e`
- `/f`

Dadurch wird die `basePath`-Eigenschaft so behandelt, als sei sie nicht vorhanden, und alle deklarierten API-Ressourcen werden relativ zum Host bereitgestellt. Dies empfiehlt sich z. B., wenn Sie einen benutzerdefinierten Domännennamen mit einem API-Mapping ohne einen Wert für den Basispfad und mit einem Wert für die Phase (die sich auf Ihre Produktionsphase bezieht) haben.

 Note

API Gateway erstellt automatisch eine Root-Ressource für Sie, auch wenn diese nicht explizit in Ihrer Definitionsdatei deklariert ist.

Wenn nicht angegeben, wird `basePath` standardmäßig auf `ignore` gesetzt.

Voranstellen

Wenn in der OpenAPI-Datei der Wert der `basePath`-Eigenschaft lautet und die `/a/b/c`-Eigenschaft `pathsund/e` enthält, führt die folgende `POST`- oder `PUT`-Anforderung:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

Dadurch wird die Eigenschaft `basePath` so behandelt, als würde sie zusätzliche Ressourcen (ohne Methoden) angeben und diese zum deklarierten Ressourcensatz hinzufügen. Dies empfiehlt sich z. B., wenn verschiedene Teams für die verschiedenen Teile einer API zuständig sind und die Eigenschaft `basePath` auf den Pfadpfadspeicherort des API-Teils der einzelnen Teams verweisen kann.

 Note

API Gateway erstellt automatisch Zwischenstufen von Ressourcen für Sie, auch wenn Sie diese nicht ausdrücklich in der Definition deklariert haben.

Teilen

Wenn in der OpenAPI-Datei der Wert der `basePath`-Eigenschaft lautet und die `/a/b/c`-Eigenschaft `paths` und `e` enthält, führt die folgende `POST`- oder `PUT`-Anforderung:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

Dadurch wird die oberste Ebene des Pfades (`/a`) als Beginn aller Pfade der Ressource behandelt und es werden zusätzliche Ressourcen (keine Methoden) in der API an sich erstellt. Das empfiehlt sich z. B., wenn `a` ein Phasenname ist, den Sie als Teil Ihrer API bereitstellen möchten.

AWS Variablen für OpenAPI-Import

Sie können die folgenden AWS Variablen in OpenAPI-Definitionen verwenden. API Gateway löst die Variablen auf, wenn die API importiert wird. Um eine Variable anzugeben, verwenden Sie `${variable-name}`. In der folgenden Tabelle werden die verfügbaren AWS Variablen beschrieben.

Variablenname	Description	
AWS::AccountId	Die AWS Konto-ID, die die API importiert. Zum Beispiel: 123456789012.	
AWS::Partition	Die AWS Partition, in die die API importiert wird. Für AWS Standardregionen ist die Partitionaws.	
AWS::Region	Die AWS Region, in die die API importiert wird. Beispiel, us-east-2 .	

AWS Beispiel für Variablen

Im folgenden Beispiel AWS werden Variablen verwendet, um eine AWS Lambda Funktion für eine Integration zu spezifizieren.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "tasks-api"
  version: "v1.0"
paths:
  /:
    get:
      summary: List tasks
      description: Returns a list of tasks
      responses:
        200:
          description: "OK"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Task"
```

```

    500:
      description: "Internal Server Error"
      content: {}
    x-amazon-apigateway-integration:
      uri:
        arn:${AWS::Partition}:apigateway:${AWS::Region}:lambda:path/2015-03-31/
        functions/arn:${AWS::Partition}:lambda:${AWS::Region}:
        ${AWS::AccountId}:function:LambdaFunctionName/invocations
      responses:
        default:
          statusCode: "200"
          passthroughBehavior: "when_no_match"
          httpMethod: "POST"
          contentHandling: "CONVERT_TO_TEXT"
          type: "aws_proxy"
  components:
    schemas:
      Task:
        type: object
        properties:
          id:
            type: integer
          name:
            type: string
          description:
            type: string

```

Fehler und Warnungen beim Import Ihrer API in API Gateway

Wenn Sie Ihre externe Definitionsdatei in API Gateway importieren, generiert API Gateway möglicherweise Warnungen und Fehler. In den folgenden Abschnitten werden die Fehler und Warnungen behandelt, die beim Import auftreten können.

Fehler beim Import

Während des Imports werden für schwerwiegende Probleme wie ungültige OpenAPI-Dokumente Fehler generiert. Fehler werden in Form von Ausnahmen als Antwort auf eine erfolglose Aktion zurückgegeben (z. B. `BadRequestException`). Wenn ein Fehler auftritt, wird die neue API-Definition verworfen und die vorhandene API wird unverändert beibehalten.

Warnungen beim Import

Während des Imports werden für kleinere Probleme wie fehlende Modellreferenzen Warnungen generiert. Falls eine Warnung auftritt, wird der Vorgang fortgesetzt, wenn an die Anforderungs-URL der Abfrageausdruck `failonwarnings=false` angehängt wird. Andernfalls werden die Aktualisierungen rückgängig gemacht. `failonwarnings` ist standardmäßig auf `false` festgelegt. In solchen Fällen werden Warnungen als Feld in der resultierenden [RestApi](#)Ressource zurückgegeben. Andernfalls werden Warnungen als Nachricht in der Ausnahme ausgegeben.

REST-API von API Gateway importieren

Sobald Sie eine REST-API in API Gateway erstellt und konfiguriert haben, können Sie sie mit der API Gateway-Konsole oder anderweitig in eine OpenAPI-Datei exportieren, und zwar mit der API Gateway-Export-API, die Teil des Amazon API Gateway-Control-Service ist. Um die API-Gateway-Export-API verwenden zu können, müssen Sie Ihre API-Anforderungen signieren. Weitere Informationen zum Signieren von Anfragen finden Sie unter [Signieren von AWS API-Anfragen](#) im IAM-Benutzerhandbuch. Ihnen stehen Optionen zur Verfügung, die API Gateway-Integrationserweiterungen sowie die [Postman](#)-Erweiterungen in die exportierte OpenAPI-Definitionsdatei aufzunehmen.

Note

Achten Sie beim Exportieren der API mithilfe von darauf AWS CLI, den Erweiterungsparameter anzugeben, wie im folgenden Beispiel gezeigt, um sicherzustellen, dass die `x-amazon-apigateway-request-validator` Erweiterung enthalten ist:

```
aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
```

Sie können eine API nicht exportieren, wenn deren Nutzlast nicht vom Typ `application/json` ist. Wenn Sie dies versuchen, erhalten Sie eine Fehlermeldung, die besagt, dass JSON-Textmodelle nicht gefunden wurden.

Anfordern zum Exportieren einer REST-API

Mit der Export-API exportieren Sie eine vorhandene REST-API, indem Sie eine GET-Anfrage einreichen und die to-be-exported API als Teil der URL-Pfade angeben. Der Anforderungs-URL hat das folgende Format:

OpenAPI 3.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/oas30
```

OpenAPI 2.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/swagger
```

Sie können die Abfragezeichenfolge `extensions` anhängen, um anzugeben, ob API Gateway-Erweiterungen (mit dem Wert `integration`) oder Postman-Erweiterungen (mit dem Wert `postman`) enthalten sein sollen.

Darüber hinaus können Sie den Header `Accept` auf `application/json` oder `application/yaml` festlegen, um die API-Definitionsausgabe im JSON- bzw. YAML-Format zu erhalten.

Weitere Informationen zum Einreichen von GET-Anfragen mit der API Gateway Export API finden Sie unter [GetExport](#).

Note

Wenn Sie Modelle in Ihrer API definieren, müssen diese für den Inhaltstyp `"application/json"` sein, damit API Gateway das Modell exportieren kann. Andernfalls löst API Gateway eine Ausnahme mit der Fehlermeldung `"Only found non-JSON body models for ..."` aus. Modelle müssen Eigenschaften enthalten oder als ein bestimmter `JSONSchema` Typ definiert sein.

Herunterladen der REST-API-OpenAPI-Definition als JSON

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen im JSON-Format herunter:

OpenAPI 3.0


```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Hier kann **<region>** beispielsweise `us-east-1` sein. Informationen zu den Regionen, in denen API Gateway verfügbar ist, finden Sie unter [Regionen und Endpunkte](#).

Herunterladen der REST-API-OpenAPI-Definition als YAML

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen im YAML-Format herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Herunterladen der REST-API-OpenAPI-Definition mit Postman-Erweiterungen als JSON

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen mit Postman im JSON-Format herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

REST-API OpenAPI-Definition mit API Gateway-Integration in YAML herunterladen

So exportieren Sie eine REST-API in OpenAPI-Definitionen mit API Gateway-Integration im YAML-Format und laden Sie herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

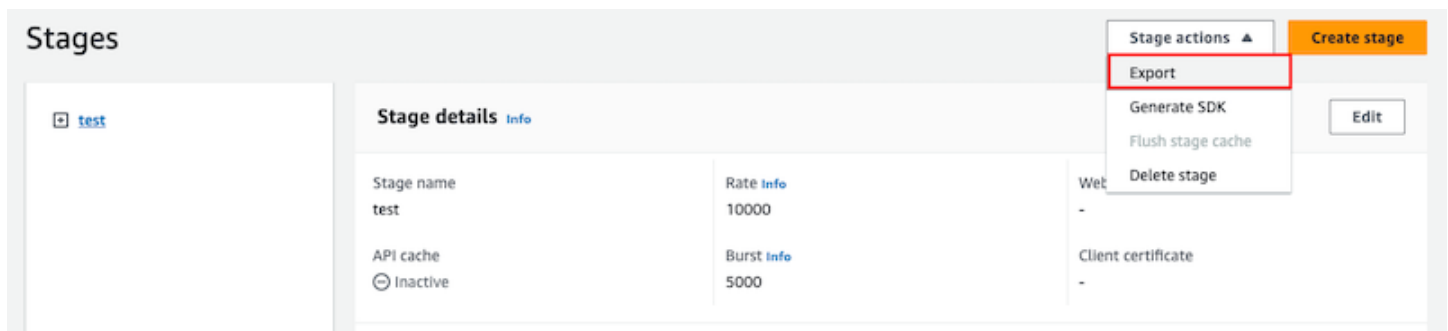
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?  
extensions=integrations  
Host: apigateway.<region>.amazonaws.com  
Accept: application/yaml
```

REST-API über die API-Gateway-Konsole exportieren

Nach der [Bereitstellung Ihrer REST-API für eine Stufe](#) können Sie mit der API Gateway-Konsole die API in der Stufe in eine OpenAPI-Datei exportieren.

Wählen Sie in der API-Gateway-Konsole im Bereich Stufen die Optionen Stufenaktionen und Export aus.



Geben Sie einen API-Spezifikationstyp, ein Format und Erweiterungen an, um die OpenAPI-Definition Ihrer API herunterzuladen.

Veröffentlichen Sie REST, APIs damit Kunden es aufrufen können

Das bloße Erstellen und Entwickeln einer API in API Gateway macht sie nicht automatisch für Ihre Benutzer aufrufbar. Um sie aufrufbar zu machen, müssen Sie Ihre API in einer Stufe bereitstellen. Darüber hinaus können Sie die URL anpassen, die Ihre Benutzer für den Zugriff auf Ihre API verwenden. Sie können ihr eine Domäne zuweisen, die mit Ihrer Marke übereinstimmt oder einprägsamer ist als die Standard-URL für Ihre API.

In diesem Abschnitt erfahren Sie, wie Sie Ihre API bereitstellen und die URL anpassen, die Sie Benutzern für den Zugriff zur Verfügung stellen.

 Note

Um die Sicherheit Ihres API Gateway zu erhöhen APIs, ist die `execute-api.`
`{region}.amazonaws.com` Domain in der [Public Suffix List \(PSL\)](#) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihr API Gateway einrichten müssen APIs. Diese Vorgehensweise trägt dazu bei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der [Set-Cookie](#)-Seite im Mozilla Developer Network.

Themen

- [REST-APIs in API Gateway bereitstellen](#)
- [Benutzerdefinierter Domainname für öffentliches REST APIs in API Gateway](#)

REST-APIs in API Gateway bereitstellen

Nach der Erstellung der API müssen Sie diese bereitstellen, damit sie von den Benutzern aufgerufen werden kann.

Zum Bereitstellen einer API erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Stufe. Eine Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. `dev`, `prod`, `beta`, `v2`). API-Stufen werden durch die API-ID und den Stufennamen identifiziert. Sie sind in der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

 Important

Jedes Mal, wenn Sie eine API aktualisieren, müssen Sie die API in einer vorhandenen Stufe oder in einer neuen Stufe erneut bereitstellen. Das Aktualisieren einer API umfasst das Ändern von Routen, Methoden, Integrationen, Genehmigern und allen anderen Details abgesehen von Stufeneinstellungen.

Wenn sich Ihre API erweitert, können Sie sie als unterschiedliche Versionen für verschiedene Stufen bereitstellen. Sie können Ihre API-Updates auch als [Canary-Release-Bereitstellung](#) bereitstellen. Auf

diese Weise können Ihre API-Clients auf derselben Stufe über die Produktionsveröffentlichung auf die Produktionsversion und über die Canary-Veröffentlichung auf die aktualisierte Version zugreifen.

Zum Aufrufen einer bereitgestellten API sendet der Client eine Anforderung mit einer API-URL. Die URL wird anhand eines API-Protokolls (HTTP (S) oder (WSS)), Hostnamens, Stufennamens und (für REST-APIs) Ressourcen-Pfads bestimmt. Der Hostname und der Stufenname bestimmen die Basis-URL der API.

Unter Verwendung des Standarddomännennamens der API weist die Basis-URL einer REST-API (beispielsweise) in einer bestimmten Stufe (*{stageName}*) das folgende Format auf:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Sie können einen benutzerdefinierten Domännennamen (z. B. *api.example.com*) erstellen und damit den standardmäßigen Hostnamen der API ersetzen, um die Basis-URL der API benutzerfreundlicher zu gestalten. Um mehrere APIs unter dem benutzerdefinierten Domännennamen zu unterstützen, müssen Sie einem Basispfad eine API-Stufe zuordnen.

Mit dem benutzerdefinierten Domännennamen *{api.example.com}* und der einem Basispfad (*{basePath}*) unter dem benutzerdefinierten Domännennamen zugeordneten API-Stufe, ändert sich die Basis-URL einer REST-API wie folgt:

```
https://{api.example.com}/{basePath}
```

Sie können für jede Stufe die API-Leistung optimieren, indem Sie die standardmäßigen Drosselungslimits für Anforderungen auf Kontoebene anpassen und das API-Caching aktivieren. Sie können auch die Protokollierung für API-Aufrufe an CloudTrail oder CloudWatch aktivieren und ein Clientzertifikat für das Backend auswählen, um die API-Anforderungen zu authentifizieren. Außerdem können Sie die Stufeneinstellungen für einzelne Methoden überschreiben und Stufenvariablen definieren, um stufenbezogenen Umgebungskontext zur Laufzeit an die API-Integration zu übergeben.

Stufen ermöglichen eine robuste Versionskontrolle Ihrer API. Sie können beispielsweise eine API für eine *test*- und eine *prod*-Stufe bereitstellen und die *test*-Stufe als Test-Build und die *prod*-Stufe als stabilen Build verwenden. Nachdem die Aktualisierungen den Test bestanden haben, können Sie die *test*-Stufe auf *prod* hochstufen. Eine Hochstufung kann durch die erneute Bereitstellung der API auf der *prod*-Stufe oder durch Aktualisieren eines Stufenvariablenwerts vom Stufennamen *test* auf *prod* erfolgen.

In diesem Abschnitt wird die Bereitstellung einer API mithilfe der [API Gateway-Konsole](#) oder durch Aufrufen der [API Gateway-REST-API](#) beschrieben. Informationen zur Verwendung anderer Tools finden Sie in der Dokumentation der [AWS CLI](#) oder eines [AWS SDK](#).

Themen

- [Eine Bereitstellung für eine REST-API in API Gateway erstellen](#)
- [Einrichten einer Stufe für eine REST-API in API Gateway](#)
- [Einrichten einer API Gateway-Canary-Release-Bereitstellung](#)
- [REST-Updates APIs , die eine erneute Bereitstellung erfordern](#)

Eine Bereitstellung für eine REST-API in API Gateway erstellen

In API Gateway wird eine REST-API-Bereitstellung durch eine [Deployment](#)-Ressource dargestellt. Sie ähnelt einer ausführbaren Datei einer API, die durch eine [RestApi](#)-Ressource dargestellt wird.

Damit der Client Ihre API aufrufen kann, müssen Sie eine Bereitstellung erstellen und ihr eine Stufe zuordnen. Eine Stufe wird durch eine [Stage](#)-Ressource dargestellt. Sie stellt einen Snapshot der API dar, einschließlich Methoden, Integrationen, Modellen, Mapping-Vorlagen und Lambda-Genehmigern (ehemals als benutzerdefinierte Genehmiger bezeichnet). Wenn Sie die API aktualisieren, können Sie sie erneut bereitstellen, indem Sie der vorhandenen Bereitstellung eine neue Stufe zuordnen. Das Erstellen einer Stufe wird in [the section called “Einrichten einer Stufe”](#) beschrieben.

Themen

- [Eine Bereitstellung auswählen](#)
- [Nächste Schritte für Ihre API-Bereitstellung](#)

Eine Bereitstellung auswählen

Im folgenden Verfahren wird gezeigt, wie Sie eine Bereitstellung für eine REST-API erstellen.

AWS-Managementkonsole

Sie müssen eine REST API erstellt haben, bevor Sie sie erstmals bereitstellen können. Weitere Informationen finden Sie unter [Entwickeln einer REST-API in API Gateway](#).

Mit der API Gateway-Konsole können Sie eine API bereitstellen, indem Sie eine Bereitstellung erstellen und sie mit einer neuen oder bestehenden Stufe verknüpfen.

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie im APIs-Navigationsbereich die API, die Sie bereitstellen möchten.
3. Wählen Sie im Bereich Ressourcen die Option API bereitstellen aus.
4. Wählen Sie für Stufe eine der folgenden Optionen aus:
 - a. Um eine neue Stufe zu erstellen, wählen Sie Neue Stufe aus und geben Sie dann einen Namen in das Feld Stufenname ein. Optional können Sie unter Beschreibung der Bereitstellung eine Beschreibung für die Bereitstellung angeben.
 - b. Um eine bestehende Stufe auszuwählen, wählen Sie den Namen der Stufe im Drop-down-Menü aus. Sie sollten unter Beschreibung der Bereitstellung eine Beschreibung der neuen Bereitstellung angeben.
 - c. Um eine Bereitstellung zu erstellen, die keiner Stufe zugeordnet ist, wählen Sie Keine Stufe aus. Später können Sie diese Bereitstellung einer Stufe zuordnen.
5. Wählen Sie Deploy (Bereitstellen) aus.

AWS CLI

Wenn Sie eine Bereitstellung erstellen, instanziiieren Sie die [Deployment](#)-Ressource.

Der folgende [create-deployment](#)-Befehl erstellt eine neue Bereitstellung:

```
aws apigateway create-deployment --rest-api-id rest-api-id
```

Sie können die API nicht aufrufen, bis Sie dieser Bereitstellung eine Stufe zugewiesen haben. Diesen Vorgang können Sie bei einer bereits vorhandenen Stufe durch Aktualisieren der Eigenschaft [deploymentId](#) der Stufe mit der neu erstellten Bereitstellungs-ID ausführen. Der folgende [update-stage](#)-Befehl aktualisiert die Stufe mit einer neuen Bereitstellung. In der Konsole wird dies als Aktive Bereitstellung bezeichnet.

```
aws apigateway update-stage \  
  --rest-api-id rest-api-id \  
  --stage-name 'stage-name' \  
  --patch-operations op='replace',path='/deploymentId',value='deployment-id'
```

Beim Erstellen Ihrer Bereitstellung können Sie diese auch gleichzeitig mit einer neuen Stufe verknüpfen. Der folgende [create-deployment](#)-Befehl erstellt eine neue Bereitstellung und verknüpft sie mit einer neuen Stufe namens beta:

```
aws apigateway create-deployment \  
  --rest-api-id rest-api-id \  
  --stage-name beta
```

Eine erneute API-Bereitstellung wird mit denselben Schritten durchgeführt. Sie können dieselbe Stufe verwenden.

Nächste Schritte für Ihre API-Bereitstellung

Im Folgenden finden Sie die nächsten Schritte für Ihre API-Bereitstellung.

Ändern der Stufeneinstellungen

Nachdem eine API bereitgestellt wurde, können Sie die Stufeneinstellungen ändern, um den API-Cache, die Protokollierung oder die Anforderungsdrosselung zu aktivieren oder zu deaktivieren. Sie können auch ein Client-Zertifikat wählen, damit das Backend API Gateway authentifizieren und Stufenvariablen festlegen kann, um den Bereitstellungskontext während der Laufzeit an die API-Integration zu übergeben. Weitere Informationen finden Sie unter [Ändern der Stufeneinstellungen](#).

Nachdem Sie die Stufeneinstellungen geändert haben, müssen Sie die API erneut bereitstellen, damit die Änderungen wirksam werden.

Note

Wenn die aktualisierten Einstellungen, wie die Aktivierung der Protokollierung, eine neue IAM-Rolle erfordern, können Sie die erforderliche IAM-Rolle ohne erneute Bereitstellung der API hinzufügen. Allerdings kann es einige Minuten dauern, bis die neue IAM-Rolle wirksam wird. Bevor dies jedoch der Fall ist, werden Spuren Ihrer API-Aufrufe nicht protokolliert, auch wenn Sie die Protokollierung aktiviert haben.

Andere Kombinationen für die Bereitstellungsstufe auswählen

Da eine Bereitstellung einen API-Snapshot darstellt und eine Stufe einen Pfad in einen Snapshot definiert, können Sie verschiedene Bereitstellungsstufen-Kombinationen wählen, um zu steuern,

wie Benutzer verschiedene Versionen der API aufrufen. Dies ist beispielsweise nützlich, wenn Sie den API-Status auf eine frühere Bereitstellung zurücksetzen oder eine "private Zweigstelle" der API in eine öffentliche integrieren möchten.

Im folgenden Verfahren wird gezeigt, wie Sie dies mithilfe des Stage Editor (Stufen-Editor) in der API Gateway-Konsole tun. Es wird davon ausgegangen, dass Sie eine API mehr als einmal bereitgestellt haben.

1. Wenn Sie sich noch nicht im Bereich Stufen befinden, wählen Sie im Hauptnavigationsbereich die Option Stufen aus.
2. Wählen Sie die Stufe aus, die Sie aktualisieren möchten.
3. Wählen Sie auf der Registerkarte Bereitstellungsverlauf die Bereitstellung aus, die die Stufe verwenden soll.
4. Wählen Sie Aktive Bereitstellung ändern.
5. Bestätigen Sie, dass Sie die aktive Bereitstellung ändern möchten, und wählen Sie Aktive Bereitstellung ändern im Dialogfeld Bereitstellung aktivieren aus.

Übergeben Sie bereitstellungsspezifische Daten an Ihre API.

Für eine Bereitstellung können Sie Stufenvariablen so festlegen oder ändern, dass bereitstellungsspezifische Daten während der Laufzeit an die API-Integration weitergegeben werden. Sie können dies auf der Registerkarte Stage Variables im Stage Editor tun. Weitere Informationen finden Sie in den Anweisungen in [Stufenvariablen für eine REST-API in API Gateway verwenden](#).

Einrichten einer Stufe für eine REST-API in API Gateway

Eine Stufe ist eine benannte Referenz auf eine Bereitstellung, bei der es sich um einen Snapshot der API handelt. Sie verwenden eine [Stufe](#), um eine bestimmte Bereitstellung zu verwalten und zu optimieren. Beispielsweise können Sie Stufeneinstellungen konfigurieren, um Zwischenspeicherung zu aktivieren, Anforderungsdrösselung anzupassen, Protokollierung zu konfigurieren, Stufenvariablen zu definieren oder ein Canary-Release für Tests anzufügen. Im folgenden Abschnitt sehen Sie, wie Sie Ihre Stufe erstellen und konfigurieren.

Erstellen einer neuen Stufe

Nach der ersten Bereitstellung können Sie weitere Stufen hinzufügen und sie mit vorhandenen Bereitstellungen verknüpfen. Sie können über die API Gateway-Konsole eine neue Stufe erstellen

oder während der Bereitstellung einer API eine vorhandene Stufe auswählen. Im Allgemeinen fügen Sie eine neue Stufe zu einer API-Bereitstellung vor der erneuten Bereitstellung der API hinzu. Gehen Sie folgendermaßen vor, um mithilfe der API Gateway-Konsole eine neue Stufe zu erstellen:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich unter APIs die Option Stages (Stufen).
4. Wählen Sie im Navigationsbereich Stages (Stufen) die Option Create stage (Stufe erstellen).
5. Geben Sie unter Stage name (Stufenname) einen Namen ein (z. B. **prod**).

 Note

Phasennamen dürfen nur alphanumerische Zeichen sowie Binde- und Unterstriche enthalten. Die maximale Länge beträgt 128 Zeichen.

6. (Optional). Geben Sie unter Description (Beschreibung) eine kurze Beschreibung der Stufe ein.
7. Wählen Sie in der Dropdown-Liste Deployment (Bereitstellung) das Datum und die Uhrzeit der vorhandenen API-Bereitstellung aus, die Sie dieser Stufe zuordnen möchten.
8. Unter Zusätzliche Einstellungen können Sie zusätzliche Einstellungen für Ihre Stufe angeben.
9. Wählen Sie Create stage (Stufe erstellen) aus.

Ändern der Stufeneinstellungen

Nach der erfolgreichen Bereitstellung einer API wird die Stufe mit Standardeinstellungen gefüllt. Sie können die Konsole oder die API Gateway-REST-API verwenden, um die Stufeneinstellungen zu ändern, einschließlich des API-Cachings und der Protokollierung. Wenn Sie den Standardendpunkt Ihrer REST-API geändert haben, wird die Änderung beim Aktualisieren einer Stufe auf alle Stufen Ihrer API übertragen. Im Folgenden wird gezeigt, wie Sie dies mithilfe des Stufen-Editor der API Gateway-Konsole durchführen.

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich unter APIs die Option Stages (Stufen).

4. Wählen Sie im Bereich Stages den Namen der Stufe aus.
5. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
6. (Optional) Bearbeiten Sie die Beschreibung in Stufenbeschreibung.
7. Ändern Sie die folgenden Einstellungen in Zusätzliche Einstellungen:

Cache-Einstellungen

Aktivieren Sie Provision-API-Cache, um API-Zwischenspeicherung für die Stufe zu erlauben. Konfigurieren Sie dann das Standard-Caching auf Methodenebene, die Cachekapazität, die Cache-Daten verschlüsseln, den Cache time-to-live (TTL) sowie alle Anforderungen für die Cache-Invalidierung pro Schlüssel.

Caching ist erst dann aktiv, wenn Sie das Standard-Caching auf Methodenebene oder den Cache auf Methodenebene für eine bestimmte Methode aktivieren.

Weitere Informationen zu Cache-Einstellungen finden Sie unter [Cache-Einstellungen für REST APIs in API Gateway](#).

Note

Wenn Sie API-Caching für eine API-Phase aktivieren, wird Ihrem AWS Konto möglicherweise das API-Caching in Rechnung gestellt. Caching kommt nicht für das kostenlose Kontingent in Frage. AWS

Drosselungseinstellungen

Aktivieren Sie die Option Drosselung zum Einrichten der Drosselungs-Targets auf Stufenebene für alle Methoden, die dieser API zugeordnet sind.

Geben Sie für Rate einen Zielsatz ein. Dies ist die Rate in Anfragen pro Sekunde, mit der Token zum Token-Bucket hinzugefügt werden. Diese Rate auf Stufenebene darf nicht größer sein als die Rate auf [Kontoebene](#), die unter [Kontingente für die Konfiguration und Ausführung einer REST-API in API Gateway](#) vorgegeben ist.

Geben Sie für Burst eine Ziel-Burst-Rate ein. Die Burst-Rate ist die Kapazität des Token-Buckets. Dies ermöglicht mehr Anfragen für einen bestimmten Zeitraum als der Zielsatz.

Diese Burst-Rate auf Stufenebene darf nicht größer sein als die Burst-Rate auf [Kontoebene](#),

die unter [Kontingente für die Konfiguration und Ausführung einer REST-API in API Gateway](#) angegeben ist.

 Note

Drosselungsraten sind keine harten Grenzwerte und werden auf Best-Effort-Basis angewendet. In manchen Fällen können Kunden die von Ihnen festgelegten Ziele überschreiten. Verlassen Sie sich nicht auf die Drosselung, um Kosten zu kontrollieren oder den Zugriff auf eine API zu blockieren. Ziehen Sie es in Betracht, [AWS Budgets](#) zu verwenden, um Kosten zu überwachen und [AWS WAF](#), um API-Anfragen zu verwalten.

Firewall- und Zertifikateinstellungen

Um der Phase eine AWS WAF Web-ACL zuzuordnen, wählen Sie eine Web-ACL aus der Dropdownliste Web-ACL aus. Falls gewünscht, wählen Sie Block API Request if WebACL cannot be evaluated (Fail- Close) (API-Anforderung blockieren, wenn WebACL nicht ausgewertet werden kann (Fehler – Schließen)).

Um ein Client-Zertifikat für Ihre Stufe auszuwählen, wählen Sie ein Zertifikat aus dem Dropdownmenü Client certificate (Client-Zertifikat) aus.

8. Klicken Sie auf Weiter.
9. Überprüfen Sie Ihre Änderungen und klicken Sie dann auf Änderungen speichern.
10. Um Amazon CloudWatch Logs für alle Methoden zu aktivieren, die mit dieser Phase dieser API-Gateway-API verknüpft sind, wählen Sie im Abschnitt Logs and tracing die Option Bearbeiten.

 Note

Um CloudWatch Logs zu aktivieren, müssen Sie auch den ARN einer IAM-Rolle angeben, die es API Gateway ermöglicht, im Namen Ihres Benutzers Informationen in CloudWatch Logs zu schreiben. Wählen Sie dazu im APIsHauptnavigationsbereich die Option Einstellungen aus. Geben Sie dann für die CloudWatch Protokollrolle den ARN einer IAM-Rolle ein.

Für allgemeine Anwendungsszenarien könnte die IAM-Rolle die verwaltete Richtlinie von [Amazon APIGateway PushToCloudWatchLogs](#) anhängen.

Die IAM-Rolle muss zudem die folgende Vertrauensstellungsanweisung enthalten:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Weitere Informationen zu CloudWatch finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

11. Wählen Sie im Dropdownmenü Protokolle eine CloudWatch Protokollierungsebene aus. Die folgenden Protokollstufen sind verfügbar:
 - Aus - Protokollierung ist für diese Stufe nicht aktiviert.
 - Nur Fehler - Protokollierung ist nur für Fehler aktiviert.
 - Fehler und Informationsprotokolle - Protokollierung ist für alle Ereignisse aktiviert.
12. Wählen Sie Datenverfolgung aus, damit API Gateway an CloudWatch die Datenverfolgungsprotokollierung für Ihre Phase berichtet. Dies kann bei der Fehlerbehebung nützlich sein APIs, kann jedoch dazu führen, dass sensible Daten protokolliert werden.

 Note

Wir empfehlen, die Datenverfolgung nicht für die Produktion APIs zu verwenden.

13. Wählen Sie Detaillierte Metriken aus, damit API Gateway über CloudWatch die API-Metriken von API calls LatencyIntegration latency,400 errors, und berichtet500 errors. Weitere Informationen zu CloudWatch finden Sie in den [Abschnitten Grundlegende Überwachung und detaillierte Überwachung](#) im CloudWatch Amazon-Benutzerhandbuch.

 Important

Ihr Konto wird für den Zugriff auf Metriken auf Methodenebene belastet, nicht jedoch für CloudWatch Metriken auf API- oder Stufenebene.

14. Aktivieren Sie die Benutzerdefinierte Zugriffsprotokollierung, um die Protokollierung für ein Ziel zuzulassen.
15. Geben Sie für ARN für Zugriffsprotokollziel den ARN einer Protokollgruppe oder eines Firehose-Streams ein.

Das ARN-Format für Firehose ist `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. Der Name Ihres Firehose-Streams muss `amazon-apigateway-{your-stream-name}` lauten.

16. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Weitere Informationen zu Beispielprotokollformaten finden Sie unter [the section called “CloudWatch Protokollformate für API Gateway”](#).
17. Wählen Sie zum Aktivieren der [AWS X-Ray](#)-Nachverfolgung für die API-Stufe die Option X-Ray tracing (X-Ray-Nachverfolgung). Weitere Informationen finden Sie unter [Benutzeranforderungen für Ablaufverfolgung an REST-APIs mithilfe von X-Ray in API Gateway](#).
18. Wählen Sie Änderungen speichern aus. Stellen Sie Ihre API erneut bereit, damit die neuen Einstellungen wirksam werden.

Überschreiben der Einstellungen auf Stufenebene

Nachdem Sie die Einstellungen auf Stufenebene angepasst haben, können Sie sie für jede API-Methode überschreiben. Bei einigen dieser Optionen können zusätzliche Gebühren für Sie anfallen AWS-Konto.

1. Vergrößern Sie die Stufe im sekundären Navigationsbereich, um Methodenüberschreibungen zu konfigurieren und wählen Sie dann eine Methode aus.

Stages

[Stage actions](#) ▼[Create stage](#)

prod

/

GET

/pets

GET

OPTIONS

POST

/{petId}

GET

OPTIONS

Method overrides

[Edit](#)

By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

 This method inherits its settings from the 'prod' stage.

Invoke URL

 <https://abcd1234.execute-api.us-west-1.amazonaws.com/prod/pets>

2. Klicken Sie für Methodenüberschreibungen auf Bearbeiten.
3. Um CloudWatch Einstellungen auf Methodenebene zu aktivieren, wählen Sie unter CloudWatch Protokolle eine Protokollierungsebene aus.
4. Wählen Sie Datenablaufverfolgung aus, um die Protokollierung der Datenablaufverfolgung für Ihre Methode zu aktivieren.

Note

Wir empfehlen, die Datenverfolgung nicht für Produktionszwecke zu verwenden. APIs

5. Klicken Sie auf Detaillierte Metriken, um diese auf Methodenebene zu aktivieren. Ihr Konto wird für den Zugriff auf Metriken auf Methodenebene belastet, nicht jedoch für CloudWatch Metriken auf API- oder Stufenebene.
6. Klicken Sie auf Drosselung, um diese auf Methodenebene zu aktivieren. Geben Sie die entsprechenden Optionen auf Methodenebene ein. Weitere Informationen zur Drosselung finden Sie unter [the section called “Drosselung”](#).
7. Klicken Sie auf Methoden-Cache aktivieren, um den Cache auf Methodenebene zu konfigurieren. Wenn Sie die Standardeinstellung für den Cache auf Methodenebene in den Stufendetails ändern, hat dies keine Auswirkungen auf diese Einstellung.
8. Wählen Sie Speichern.

Fügen Sie eine API-Gateway-REST-API als Ziel für Amazon Bedrock AgentCore Gateway hinzu

Ein Amazon Bedrock AgentCore Gateway bietet Entwicklern von KI-Agenten eine sichere Möglichkeit, Ihren API-Gateway-REST APIs als Model Context Protocol (MCP) -kompatible Tools bereitzustellen. AgentCore Gateway verwendet Ziele, um Tools zu definieren. Wenn Sie Ihre Phase als Ziel hinzufügen, wird Ihr Gateway zu einer einzigen MCP-URL, die einem Agenten den Zugriff auf die Tools ermöglicht. Weitere Informationen finden Sie unter [API Gateway REST API-Stufen als Ziele](#) im Amazon Bedrock AgentCore Gateway Developer Guide.

API-Gateway-Ziele verbinden Ihr AgentCore Gateway mit Phasen Ihres REST APIs. Sie können die gesamte Phase als Ziel einbeziehen oder Ressourcen auswählen. Nachdem Sie das API-Gateway-Ziel erstellt haben, übersetzt AgentCore Gateway eingehende MCP-Anfragen in HTTP-Anfragen und kümmert sich um die Formatierung der Antwort. MCP-Clients können die API-Dokumentation mithilfe der `tools/list` Methode abrufen und mithilfe der Methode aufrufen APIs . `tools/call`

Überlegungen

Die folgenden Überlegungen können sich auf Ihre Verwendung auswirken, wenn Sie einem Gateway eine Phase als Ziel hinzufügen: AgentCore

- Sie müssen bereits über ein AgentCore Gateway verfügen.
- Nur öffentliche APIs REST-Dateien werden unterstützt.
- Der Standardendpunkt Ihrer API kann nicht deaktiviert werden.
- Für jede Methode Ihrer API muss entweder ein [Operationsname](#) definiert sein, oder Sie müssen eine Namensüberschreibung erstellen, wenn Sie Ihre Phase als Ziel hinzufügen. Dieser Name wird als Name des Tools verwendet, mit dem Agenten mit Ihrer Methode interagieren.

- Sie können die Typen `API_KEY`, `NO_AUTH`, oder `GATEWAY_IAM_ROLE` Credential Provider für Outbound Auth verwenden, um Ihrem Gateway den Zugriff auf Ihre API zu ermöglichen. Der Anbieter für `API_KEY` Anmeldeinformationen wird von Gateway definiert. AgentCore Sie können Ihren vorhandenen API-Gateway-API-Schlüssel verwenden. Weitere Informationen finden Sie unter [Outbound Auth einrichten](#).
- Wenn Sie einen Amazon Cognito Cognito-Benutzerpool oder einen Lambda-Autorisierer verwenden, um den Zugriff auf Ihre API zu kontrollieren, können MCP-Clients nicht darauf zugreifen.
- Ihre API muss sich im selben Konto und in derselben Region wie Ihr Gateway befinden. AgentCore

Fügen Sie eine API-Stufe als Ziel für ein AgentCore Gateway hinzu

Das folgende Verfahren zeigt, wie Sie eine API-Stufe als Ziel für ein AgentCore Gateway hinzufügen.

So fügen Sie eine API-Stufe als Ziel für ein AgentCore Gateway hinzu

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus, die in einer Phase bereitgestellt wird.
3. Klicken Sie im Hauptnavigationsbereich auf Stufen.
4. Wählen Sie Stage-Aktionen und dann Create MCP-Ziel aus.
5. Wählen Sie für AgentCore Gateway ein AgentCore Gateway aus.
6. Geben Sie unter Zielname einen Zielnamen ein.
7. Geben Sie unter Zielbeschreibung eine Beschreibung ein.
8. Behalten Sie die bereitgestellte API und die bereitgestellte Phase bei.
9. Wählen Sie unter API-Ressourcen auswählen die Ressourcen Ihrer API aus, auf die Agenten, die Ihr AgentCore Gateway verwenden, zugreifen können.

Wenn Sie keine Ressource auswählen, kann ein Agent die Dokumentation nicht einsehen oder den Endpunkt aufrufen.

10. Die Kombination aus Ressource und Methode sind die Operationen für das Tool. Wenn Ihre Operation keinen Namen hat, erstellen Sie eine Namensüberschreibung.

Sie können auch einen Operationsnamen für eine Methode definieren, wenn Sie sie erstellen.

11. Wählen Sie für die Konfiguration „Outbound Auth“ entweder „IAM-Rolle“, „Keine Autorisierung“ oder „API-Schlüssel“.

12. Wählen Sie Ziel erstellen.

Um alle AgentCore Gateways anzuzeigen, die Zugriff auf Ihre haben APIs, wählen Sie im Hauptnavigationsbereich den Abschnitt MCP-Ziele aus. In diesem Abschnitt können Sie ein MCP-Ziel für jede API in Ihrer Region erstellen, die in einer Phase bereitgestellt wird. Wählen Sie Create MCP-Ziel aus und folgen Sie den vorherigen Schritten.

Sie können auch die verfügbaren Tools für Ihr Ziel anzeigen und Ihr Ziel in der AgentCore Gateway-Konsole bearbeiten. Weitere Informationen finden [Sie unter Ziele zu einem vorhandenen AgentCore Gateway hinzufügen](#).

Löschen einer Stufe

Wenn Sie eine Stufe nicht mehr benötigen, können Sie diese löschen, um Kosten für ungenutzte Ressourcen zu vermeiden. In den folgenden Schritten wird gezeigt, wie Sie die API Gateway-Konsole zum Löschen einer Stufe verwenden.

Warning

Das Löschen einer Stufe kann dazu führen, dass ein Teil der oder die gesamte zugehörige API nicht mehr von API-Aufrufern verwendet werden kann. Das Löschen einer Stufe kann nicht rückgängig gemacht werden. Sie können die Stufe jedoch neu erstellen und derselben Bereitstellung zuordnen.

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
4. Wählen Sie im Bereich Stages (Stufen) die Stufe aus, die Sie löschen möchten, und klicken Sie dann unter Stage actions (Stufenaktionen) auf Delete stage (Stufe löschen).
5. Geben Sie bei der Aufforderung **confirm** ein und klicken Sie dann auf Delete (Löschen).

Tags für eine API-Stufe in API Gateway einrichten

In API Gateway können Sie einer API-Stufe ein Tag hinzufügen, das Tag aus der Stufe entfernen oder das Tag anzeigen. Dazu können Sie die API-Gateway-Konsole, die AWS CLI/das SDK oder die API-Gateway-REST-API verwenden.

Eine Stufe kann auch Tags von der übergeordneten REST-API erben. Weitere Informationen finden Sie unter [the section called “Tag-Vererbung in der Amazon API Gateway V1-API”](#).

Weitere Informationen über das Markieren mit Tags von API Gateway-Ressourcen finden Sie unter [Markieren](#).

Themen

- [Einrichten von Tags für eine API-Stufe mit der API-Gateway-Konsole](#)
- [Tags für eine API-Stufe mithilfe der AWS CLI-Konsole einrichten](#)
- [Einrichten von Tags für eine API-Stufe mit der API-Gateway-REST-API](#)

Einrichten von Tags für eine API-Stufe mit der API-Gateway-Konsole

Die folgende Vorgehensweise beschreibt, wie Sie Tags für eine API-Stufe einrichten.

So richten Sie Tags für eine API-Stufe mithilfe der API Gateway-Konsole ein

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene API aus oder erstellen Sie eine neue API, die Ressourcen, Methoden und die entsprechenden Integrationen enthält.
3. Wählen Sie eine Stufe aus oder stellen Sie die API in einer neuen Stufe bereit.
4. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
5. Wählen Sie die Registerkarte Tags aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
6. Wählen Sie Tags verwalten aus.
7. Klicken Sie im Tag Editor (Tag-Editor) auf Add Tag (Tag hinzufügen). Geben Sie einen Tag-Schlüssel (z. B. Department) in das Feld Schlüssel und einen Tag-Wert (z. B. Sales) in das Feld Wert ein. Wählen Sie Save (Speichern) aus, um das Tag zu speichern.
8. Wiederholen Sie bei Bedarf Schritt 5, um der API-Stufe weitere Tags hinzuzufügen. Die maximale Anzahl der Tags pro Stufe beträgt 50.

9. Um ein vorhandenes Tag aus der Stufe zu entfernen, wählen Sie Remove (Entfernen) aus.
10. Wenn die API zuvor in der API Gateway-Konsole bereitgestellt wurde, müssen Sie sie erneut implementieren, damit die Änderungen wirksam werden.

Tags für eine API-Stufe mithilfe der AWS CLI-Konsole einrichten

Sie können Tags für eine API-Stufe mithilfe von AWS CLI und dem Befehl [create-stage](#) oder [tag-resource](#) erstellen. Sie können einen oder mehrere Tags aus einer API-Stufe mithilfe des Befehls [untag-resource](#) löschen.

Der folgende [create-stage](#)-Befehl fügt beim Erstellen einer test-Stufe ein Tag hinzu:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description  
'Testing stage' --deployment-id efg456 --tag Department=Sales
```

Der folgende [tag-resource](#)-Befehl fügt einer prod-Stufe ein Tag hinzu:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/  
restapis/abc123/stages/prod --tags Department=Sales
```

Der folgende [untag-resource](#)-Befehl entfernt das Tag Department=Sales aus der test-Stufe:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/  
restapis/abc123/stages/test --tag-keys Department
```

Einrichten von Tags für eine API-Stufe mit der API-Gateway-REST-API

Sie können Tags für eine API-Stufe mit Hilfe der API Gateway-REST-API einrichten, indem Sie eine der folgenden Aktionen ausführen:

- Rufen Sie [tags:tag](#) auf, um eine API-Stufe mit einem Tag zu versehen.
- Rufen Sie [tags:untag](#) auf, um einen oder mehrere Tags aus einer API-Stufe zu löschen.
- Rufen Sie [stage:create](#) auf, um einen oder mehrere Tags zu einer API-Stufe, die Sie erstellen, hinzuzufügen.

Sie können außerdem [tags:get](#) aufrufen, um Tags in einer API-Stufe zu beschreiben.

Tag zu einer API-Stufe hinzufügen

Nach Bereitstellung einer API (m5zr3vnks7) für eine Stufe (test) versehen Sie die Stufe durch Aufruf von [tags:tag](#) mit einem Tag. Die erforderliche Amazon-Ressourcenname (ARN) (arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test) der Stufe muss URL-codiert sein (arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest).

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest

{
  "tags" : {
    "Department" : "Sales"
  }
}
```

Sie können auch die vorherige Anforderung verwenden, um einen bestehenden Tag auf einen neuen Wert zu aktualisieren.

Sie können Tags zu einer Stufe hinzufügen, wenn Sie [stage:create](#) aufrufen, um die Stufe zu erstellen:

```
POST /restapis/<restapi_id>/stages

{
  "stageName" : "test",
  "deploymentId" : "adr134",
  "description" : "test deployment",
  "cacheClusterEnabled" : "true",
  "cacheClusterSize" : "500",
  "variables" : {
    "sv1" : "val1"
  },
  "documentationVersion" : "test",

  "tags" : {
    "Department" : "Sales",
    "Division" : "Retail"
  }
}
```

Tag von einer API-Stufe entfernen

Rufen Sie [Department](#) auf, um den `tags:untag`-Tag von der Stufe zu entfernen:

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Um mehrere Tags zu entfernen, verwenden Sie eine kommasetrennte Liste mit Tag-Schlüsseln im Abfrageausdruck (zum Beispiel `?tagKeys=Department,Division,...`).

Beschreiben von Tags für eine API-Stufe

Um existierende Tags einer bestehenden Stufe zu beschreiben, rufen Sie [tags:get](#) auf:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
200 OK

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/
restapi-tags-{rel}.html",
      "name": "tags",
      "templated": true
    },
    "tags:tag": {
      "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftags"
    },
    "tags:untag": {
      "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftags{?tagKeys}",
      "templated": true
    }
  }
}
```

```
    },  
    "tags": {  
        "Department": "Sales"  
    }  
}
```

Stufenvariablen für eine REST-API in API Gateway verwenden

Bei Stufenvariablen handelt es sich um Wert-Paare, die Sie als Konfigurationsattribute für eine REST-API-Bereitstellungsstufe definieren können. Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung und Mapping-Vorlagen verwendet werden. Mit den Bereitstellungsstufen in API Gateway können Sie für jede API mehrere Freigabestufen verwalten. Über die Stufenvariablen können Sie eine API-Bereitstellungsstufe für die Interaktion mit verschiedenen Backend-Endpunkten konfigurieren.

Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Generators übergeben. Weitere Informationen hierzu finden Sie unter [the section called “Ausgabe von einem API-Gateway-Lambda-Generator”](#).

Anwendungsfälle für Stufenvariablen

Im Folgenden sehen Sie Anwendungsfälle für Ihre Stufenvariablen.

Geben Sie einen anderen Backend-Endpunkt an

Ihre API kann beispielsweise eine GET-Anforderung als HTTP-Proxy an den Backend-Webhost übergeben. Mit einer Stufenvariable können Sie veranlassen, dass API Gateway `example.com` aufruft, wenn API-Aufrufer Ihren Produktionsendpunkt aufrufen. Wenn API-Aufrufer dann die Beta-Phase aufrufen, ruft API Gateway einen anderen Webhost auf, z. B. `beta.example.com`. Ebenso lässt sich über Stufenvariablen für jede Stufe Ihrer API ein anderer AWS Lambda - Funktionsname angeben. Sie können eine Stufenvariable nicht für die Festlegung eines anderen Integrationsendpunkts verwenden, z. B. um die GET-Anforderung auf eine HTTP-Proxyintegration in einer Stufe, und eine Lambda-Proxyintegration in einer anderen Stufe zu verweisen.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Wenn Sie eine Lambda-Funktion in der API Gateway Gateway-Konsole angeben, erscheint ein AWS CLI Befehl zur Konfiguration der richtigen Berechtigungen. Sie können dazu auch den folgenden AWS CLI Befehl verwenden.

```
aws lambda add-permission --function-name "arn:aws:lambda:us-east-2:123456789012:function:my-function" --source-arn "arn:aws:execute-api:us-east-2:123456789012:api_id/*/HTTP_METHOD/resource" --principal apigateway.amazonaws.com --statement-id apigateway-access --action lambda:InvokeFunction
```

Informationen mithilfe von Zuweisungsvorlagen übergeben

Sie können in den Zuweisungsvorlagen auf Stufenvariablen zugreifen oder Konfigurationsparameter an Ihr AWS Lambda oder das HTTP-Backend übergeben. Beispielsweise möchten Sie vielleicht die gleiche Lambda-Funktion für mehrere Stufen in Ihrer API verwenden, jedoch soll die Funktion abhängig von der aufgerufenen Stufe jeweils Daten aus einer anderen Amazon DynamoDB-Tabelle auslesen. In den Mapping-Vorlagen, von denen die Anforderung für die Lambda-Funktion generiert wird, können Sie den Tabellennamen mit Stufenvariablen an Lambda übergeben.

Eine Stufenvariable muss zunächst konfiguriert und dann muss dieser ein Wert zugewiesen werden, bevor Sie die Stufenvariable verwenden können. Um beispielsweise den HTTP-Integrationsendpunkt anzupassen, erstellen Sie zuerst die `url`-Stufenvariable und geben dann den Wert der Stufenvariablen (**`http://${stageVariables.url}`**) in der Integrationsanforderung Ihrer API ein. Mit diesem Wert wird API Gateway angewiesen, die Stufenvariable `${}` zur Laufzeit abhängig von der ausgeführten API-Stufe zu ersetzen. Weitere Informationen finden Sie unter [the section called “Stufenvariablen für REST APIs in API Gateway einrichten”](#).

Stufenvariablen für REST APIs in API Gateway einrichten

In diesem Abschnitt erfahren Sie, wie Sie mit der Amazon-API-Gateway-Konsole verschiedene Stufenvariablen für zwei Bereitstellungsstufen einer Beispiel-API einrichten. Für ein besseres Verständnis, wie Stufenvariablen in API Gateway verwendet werden, empfehlen wir, dass Sie alle Schritte in diesem Abschnitt befolgen.

Voraussetzungen

Überprüfen Sie vor Beginn, ob die folgenden Anforderungen erfüllt sind:

- Es muss eine API in API Gateway verfügbar sein. Folgen Sie den Anweisungen in [Entwickeln einer REST-API in API Gateway](#).
- Sie müssen die API mindestens einmal bereitgestellt haben. Folgen Sie den Anweisungen in [REST-APIs in API Gateway bereitstellen](#).

- Sie müssen die erste Stufe für eine bereitgestellte API erstellt haben. Folgen Sie den Anweisungen in [Erstellen einer neuen Stufe](#).

Aufrufen eines HTTP-Endpunkts über eine API mit einer Stufenvariable

In diesem Verfahren wird beschrieben, wie Sie eine Stufenvariable für einen HTTP-Endpunkt und zwei Stufen für Ihre API erstellen. Darüber hinaus erstellen Sie die Stufenvariablen `url`, `stageName` und `function`, die in den folgenden Abläufen in diesem Abschnitt verwendet werden.

Aufrufen eines HTTP-Endpunkts über eine API mit einer Stufenvariable

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Erstellen Sie eine API und anschließend eine GET-Methode für die Root-Ressource der API. Stellen Sie den Integrationstyp auf HTTP und die Endpunkt-URL auf **`http://${stageVariables.url}`**.
3. Stellen Sie die API für eine neue Stufe namens **beta** bereit.
4. Wählen Sie im Navigationsbereich Stages (Stufen) die Stufe beta.
5. Wählen Sie auf der Registerkarte Stage variables (Stufenvariablen) die Option Edit (Bearbeiten) aus.
6. Wählen Sie Add stage variable (Stufenvariable hinzufügen).
7. Geben Sie unter Name **url** ein. Geben Sie für Value (Wert) **httpbin.org/get** ein.
8. Wählen Sie Add stage variable (Stufenvariable hinzufügen) und gehen Sie wie folgt vor:

Geben Sie unter Name **stageName** ein. Geben Sie für Value (Wert) **beta** ein.
9. Wählen Sie Add stage variable (Stufenvariable hinzufügen) und gehen Sie wie folgt vor:

Geben Sie unter Name **function** ein. Geben Sie für Value (Wert) **HelloWorld** ein.
10. Wählen Sie Speichern.
11. Erstellen Sie nun eine zweite Stufe. Wählen Sie im Navigationsbereich Stages (Stufen) die Option Create stage (Stufe erstellen). Geben Sie für Stage name (Stufenname) **prod** ein. Wählen Sie in Deployment (Bereitstellung) eine kürzlich ausgeführte Bereitstellung und anschließend Create stage (Stufe erstellen) aus.

12. Legen Sie wie im Fall der Phase beta die gleichen drei Stufenvariablen (url, stageName und function) auf unterschiedliche Werte fest („**petstore-demo-endpoint.execute-api.com/petstore/pets**“, „**prod**“ und „**HelloEveryone**“).
13. Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hierdurch wird die beta-Stufen-Anforderung GET für die Stammressource der API gestartet.

 Note

Der Link Invoke URL verweist auf die Stammressource der API auf der beta-Stufe. Durch die Eingabe der URL in einem Webbrowser wird die beta-Stufenmethode GET für die Stammressource aufgerufen. Wenn Methoden für untergeordnete Ressourcen und nicht für die Stammressource selbst definiert sind, erhalten Sie bei der Eingabe der URL in einen Webbrowser die Fehlermeldung `{"message": "Missing Authentication Token"}`. In diesem Fall müssen Sie dem Link Invoke URL den Namen einer spezifischen untergeordneten Ressource anfügen.

14. Im Folgenden wird die Antwort angezeigt, die Sie von der beta-Stufen-Anforderung GET erhalten. Sie können das Ergebnis auch überprüfen, indem Sie in einem Browser zu `http://httpbin.org/get` navigieren. Dieser Wert wurde der `url`-Variablen in der beta-Stufe zugewiesen. Die beiden Antworten sind identisch.
15. Wählen Sie im Navigationsbereich Stages die Stufe prod. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hierdurch wird die prod-Stufen-Anforderung GET für die Stammressource der API gestartet.
16. Im Folgenden wird die Antwort angezeigt, die Sie von der prod-Stufen-Anforderung GET erhalten. Sie können das Ergebnis überprüfen, indem Sie in einem Browser zu `http://.execute-api.petstore-demo-endpoint.com/petstore/pets` navigieren. Dieser Wert wurde der `url`-Variablen in der prod-Stufe zugewiesen. Die beiden Antworten sind identisch.

Übergabe phasenspezifischer Metadaten an ein HTTP-Backend

In diesem Verfahren wird beschrieben, wie stufenbezogene Metadaten mittels eines Stufenvariablenwerts in einem Abfrageparameterausdruck an ein HTTP-Backend übergeben werden.

Hier verwenden wir die `stageName`-Stufenvariable, die im vorangegangenen Verfahren deklariert wurde.

Übergabe phasenspezifischer Metadaten an ein HTTP-Backend

1. Wählen Sie im Navigationsbereich Resource die Methode GET.

Um der URL der Methode einen Abfragezeichenfolge-Parameter hinzuzufügen, wählen Sie die Registerkarte Method Request (Methodenanforderung) und anschließend im Abschnitt Method request settings (Einstellungen der Methodenanforderung) die Option Edit (Bearbeiten).

2. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
 - b. Geben Sie unter Name **`stageName`** ein.
 - c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
3. Wählen Sie Speichern.
4. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Edit (Bearbeiten) aus.
5. Fügen Sie bei Endpoint URL (Endpunkt-URL) den Wert **`?stageName=${stageVariables.stageName}`** an den zuvor definierten URL-Wert an, damit die gesamte Endpoint URL (Endpunkt-URL) **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`** lautet.
6. Wählen Sie Deploy API (API bereitstellen) und wählen Sie die Stufe beta aus.
7. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen). Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein.

Note

Sie verwenden hier die beta-Stufe, da der HTTP-Endpunkt (von der `url`-Variable mit "http://httpbin.org/get" angegeben) Abfrageparameterausdrücke annimmt und diese in der Antwort als `args`-Objekt zurückgibt.

8. Sie erhalten die folgende Antwort. Beachten Sie, dass beta (zugeordnet zur stageName-Stufenvariable) als stageName-Argument an das Backend übergeben wird.

```
{
  "args": {
    "stageName": "beta"
  },
  "headers": {
    "Accept": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "AmazonAPIGateway_abcd1234",
    "X-Amzn-ApiGateway-API-Id": "abcd1234",
    "X-Amzn-Trace-Id": "Self=1-abcd-11111111111111;Root=1-11111111-1111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Aufrufen einer Lambda-Funktion über eine API mit einer Stufenvariable

In diesem Verfahren wird beschrieben, wie Sie eine Stufenvariable verwenden, um eine Lambda-Funktion als Backend der API aufzurufen. Hier verwenden Sie die function-Stufenvariable, die in [Aufrufen eines HTTP-Endpunkts über eine API mit einer Stufenvariable](#) deklariert wurde.

Wenn Sie eine Lambda-Funktion als Wert einer Stufenvariable festlegen, verwenden Sie den lokalen Namen der Funktion und binden den Alias oder die Versionsangabe ein, z. B. **HelloWorld**, **HelloWorld:1** oder **HelloWorld:alpha**. Nutzen Sie nicht den ARN der Funktion (z. B. **arn:aws:lambda:us-east-1:123456789012:function:HelloWorld**). Die API Gateway-Konsole betrachtet den Stufenvariablenwert einer Lambda-Funktion als unvollständigen Funktionsnamen und erweitert die Stufenvariable zu einem ARN.

Aufrufen einer Lambda-Funktion über eine API mit einer Stufenvariable

1. Erstellen Sie eine Lambda-Funktion namens **HelloWorld** mit der Laufzeit Node.js. Der Code muss Folgendes enthalten:

```
export const handler = function(event, context, callback) {
  if (event.stageName)
    callback(null, 'Hello, World! I\'m calling from the ' + event.stageName + ' stage.');
```

```
    else
      callback(null, 'Hello, World! I\'m not sure where I\'m calling from...');
```

```
};
```

Weitere Informationen zum Erstellen einer Lambda-Funktion finden Sie unter [Erste Schritte mit der REST-API-Konsole](#).

2. Wählen Sie im Bereich Ressourcen (Ressourcen) die Option Create resource (Ressource erstellen) aus, und gehen Sie dann wie folgt vor:
 - a. Wählen Sie für Resource path (Ressourcenpfad) die Option/aus.
 - b. Geben Sie für Resource name (Ressourcenname) **lambdav1** ein.
 - c. Wählen Sie Create Resource (Ressource erstellen) aus.
3. Wählen Sie die /lambdav1-Ressource aus und klicken Sie dann auf Methode erstellen.

Führen Sie dann die folgenden Schritte aus:

- a. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
- b. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
- c. Lassen Sie Lambda proxy integration (Lambda-Proxyintegration) deaktiviert.
- d. Geben Sie für die Lambda function (Lambda-Funktion) `${stageVariables.function}` ein.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▼

✕

Tip

Wenn Sie zur Eingabe des Berechtigungsbehl hinzufügen aufgefordert werden, kopieren Sie den [add-permission](#)-Befehl. Sie müssen den Befehl für jede Lambda-Funktion ausführen, die der function-Stufenvariable zugewiesen werden soll. Wenn der `$stageVariables.function`-Wert beispielsweise HelloWorld lautet, führen Sie den folgenden AWS CLI -Befehl aus:

```
aws lambda add-permission --function-name arn:aws:lambda:us-east-1:account-id:function:HelloWorld --source-arn arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/lambdav1 --principal
```

```
apigateway.amazonaws.com --statement-id statement-id-guid --action
lambda:InvokeFunction
```

Falls das nicht geschieht, führt der Aufruf der Methode zur Fehlermeldung 500 Internal Server Error. Ersetzen Sie `${stageVariables.function}` durch den Lambda-Funktionsnamen, der dieser Stufenvariable zugeordnet ist.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1

Q `${stageVariables.function}`



⚠ You defined your Lambda function as a stage variable. Run the following AWS CLI command to ensure you have the appropriate policy for this function. Replace the stage variable in the function-name parameter with the necessary function name.

▼ Add permission command

```
1 aws lambda add-permission \
2 --function-name "arn:aws:lambda:us-east-1:111122223333:function:${stageVariables.function}" \
3 --source-arn "arn:aws:execute-api:us-east-1:111122223333:abcd1234/*/GET/lambda1" \
4 --principal apigateway.amazonaws.com \
5 --statement-id abcd-12345-efg \
6 --action lambda:InvokeFunction
```



- e. Wählen Sie Methode erstellen aus.
4. Stellen Sie die API in den Stufen prod und beta bereit.
5. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen). Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hängen Sie **/lambda1** an die URL an, bevor Sie die Eingabetaste drücken.

Sie erhalten die folgende Antwort.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Übergeben von stufenbezogenen Metadaten an eine Lambda-Funktion über eine Stufenvariable

In diesem Verfahren wird beschrieben, wie stufenbezogene Konfigurationsmetadaten mittels einer Stufenvariable an eine Lambda-Funktion übergeben werden. Erstellen Sie eine POST-Methode und

eine Eingabe-Mapping-Vorlage, um Nutzlast mit der zuvor deklarierten `stageName`-Stufenvariable zu generieren.

Übergeben von stufenbezogenen Metadaten an eine Lambda-Funktion über eine Stufenvariable

1. Wählen Sie die `/lambdav1`-Ressource aus und klicken Sie dann auf **Methode erstellen**.

Führen Sie dann die folgenden Schritte aus:

- a. Wählen Sie unter **Method type** (Methodentyp) die Option **POST** aus.
 - b. Wählen Sie für den **Integration type** (Integrationstyp) die Option **Lambda function** (Lambda-Funktion) aus.
 - c. Lassen Sie **Lambda proxy integration** (Lambda-Proxyintegration) deaktiviert.
 - d. Geben Sie für die **Lambda function** (Lambda-Funktion) `${stageVariables.function}` ein.
 - e. Wenn Sie zur Eingabe des Berechtigungsbefehls hinzugefügt aufgefordert werden, kopieren Sie den [add-permission](#)-Befehl. Sie müssen den Befehl für jede Lambda-Funktion ausführen, die der `function`-Stufenvariable zugewiesen werden soll.
 - f. Wählen Sie **Methode erstellen** aus.
2. Wählen Sie die Registerkarte **Integration request** (Integrationsanforderung) und dann im Abschnitt **Integration request settings** (Einstellungen für Integrationsanforderungen) die Option **Edit** (Bearbeiten) aus.
 3. Wählen Sie Mapping-Vorlagen aus und klicken Sie dann auf **Mapping-Vorlage hinzufügen**.
 4. Geben Sie für **Content type** (Inhaltstyp) **application/json** ein.
 5. Geben Sie für **Template body** (Vorlagentext) die folgende Vorlage ein:

```
#set($inputRoot = $input.path('$'))
{
  "stageName" : "$stageVariables.stageName"
}
```

 Note

In einer Mapping-Vorlage muss eine Stufenvariable in Anführungszeichen (wie in `"$stageVariables.stageName"` oder `"${stageVariables.stageName}"`)

referenziert werden. An anderen Stellen muss sie ohne Anführungszeichen (wie in) referenziert werden `${stageVariables.function}`).

6. Wählen Sie Speichern.
7. Stellen Sie die API in den Stufen beta und prod bereit.
8. Gehen Sie wie folgt vor, um einen REST-API-Client zum Übergeben von stufenspezifischen Metadaten zu verwenden:
 - a. Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in das Eingabefeld eines REST-API-Clients ein. Hängen Sie **/lambda1** an, bevor Sie Ihre Anfrage absenden.

Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the beta stage."
```

- b. Wählen Sie im Navigationsbereich Stages (Stufen) die Option prod. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in das Eingabefeld eines REST-API-Clients ein. Hängen Sie **/lambda1** an, bevor Sie Ihre Anfrage absenden.

Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the prod stage."
```

9. Gehen Sie wie folgt vor, um die Funktion Test zum Übergeben von stufenspezifischen Metadaten zu verwenden:
 - a. Wählen Sie im Navigationsbereich Resources (Ressourcen) die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
 - b. Geben Sie für function **HelloWorld** ein.
 - c. Geben Sie für stageName **beta** ein.
 - d. Wählen Sie Test aus. Sie müssen Ihrer POST-Anfrage keinen Text hinzufügen.

Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the beta stage."
```


- e. Sie können die vorherigen Schritte wiederholen, um die Prod-Stufe zu testen. Geben Sie für `stageName` **Prod** ein.

Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the prod stage."
```

API Gateway-Stufenvariablenreferenz für REST APIs in API Gateway

Sie können die API Gateway-Stufenvariablen in folgenden Fällen verwenden.

Parameter-Mapping-Ausdrücke

Eine Stufenvariable kann in einem Parameter-Mapping-Ausdruck für den Header-Parameter einer API-Methodenanforderung oder -antwort genutzt werden, und zwar ohne Teilsubstitution. Im folgenden Beispiel wird die Stufenvariable ohne `$` und umschließende `{ . . . }` referenziert.

- `stageVariables.<variable_name>`

Mapping-Vorlagen

Eine Stufenvariable kann überall in einer Mapping-Vorlage verwendet werden, wie in den folgenden Beispielen dargestellt.

- `{ "name" : "$stageVariables.<variable_name>" }`
- `{ "name" : "${stageVariables.<variable_name>}" }`

HTTP-Integration URIs

Eine Stufenvariable kann als Teil einer HTTP-Integrations-URL verwendet werden, wie in den folgenden Beispielen dargestellt:

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`

- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS Integration URIs

Eine Stufenvariable kann als Teil von AWS URI-Aktions- oder Pfadkomponenten verwendet werden, wie im folgenden Beispiel gezeigt.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS Integration URIs (Lambda-Funktionen)

Eine Stufenvariable kann anstelle des Lambda-Funktionsnamens (oder Version/Alias) verwendet werden, wie in den folgenden Beispielen dargestellt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

Note

Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

Amazon-Cognito-Benutzerpool

Eine Stufenvariable kann anstelle eines Amazon-Cognito-Benutzerpools für einen COGNITO_USER_POOLS-Genehmiger verwendet werden.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWS Anmeldeinformationen für die Integration

Eine Stufenvariable kann als Teil des ARN für AWS Benutzer-/Rollenanmeldedaten verwendet werden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Einrichten einer API Gateway-Canary-Release-Bereitstellung

[Canary-Release](#) ist eine Softwareentwicklungsstrategie, bei der eine neue Version einer API (und anderer Software) zu Testzwecken eingesetzt wird und die Basisversion weiterhin als Produktionsrelease für den normalen Betrieb auf derselben Stufe eingesetzt wird. Im Rahmen dieses Artikels gehen wir davon aus, dass die Basisversion das Produktionsrelease ist. Sie können das Canary-Release jedoch auch für Tests in jeder anderen Version verwenden.

Bei einer Canary-Release-Bereitstellung wird der gesamte API-Verkehr mit einem vorkonfigurierten Verhältnis nach dem Zufallsprinzip in ein Produktionsrelease und ein Canary-Release getrennt. Normalerweise erhält die Canary-Release einen kleinen Prozentsatz des API-Verkehrs und das Produktionsrelease nimmt den Rest in Anspruch. Die aktualisierten API-Features sind nur für den API-Verkehr über das Canary-Release sichtbar. Sie können den Prozentsatz des Canary-Datenverkehrs anpassen, um die Testabdeckung oder die Leistung zu optimieren.

Indem man den Canary-Verkehr gering und die Auswahl zufällig hält, werden die meisten Benutzer zu keiner Zeit durch potenzielle Fehler in der neuen Version beeinträchtigt und kein einzelner Benutzer wird ständig beeinträchtigt.

Nachdem die Testmetriken Ihre Anforderungen erfüllt haben, können Sie das Canary-Release in das Produktionsrelease überführen und Canary für die Bereitstellung deaktivieren. Damit stehen die neuen Features bereits in der Produktionsstufe zur Verfügung.

Themen

- [Canary-Release-Bereitstellung in API Gateway](#)
- [Erstellen einer Canary-Release-Bereitstellung](#)
- [Aktualisieren eines Canary-Releases](#)
- [Hochstufen eines Canary-Releases](#)
- [Deaktivieren eines Canary-Releases](#)

Canary-Release-Bereitstellung in API Gateway

In API Gateway verwendet eine Canary-Release-Bereitstellung die Bereitstellungsstufe für die Produktionsfreigabe der Basisversion einer API und fügt der Stufe ein Canary-Release für die neuen Versionen relativ zur Basisversion der API hinzu. Die Stufe ist der ersten Bereitstellung und das Canary-Release den nachfolgenden Bereitstellungen zugeordnet. Zu Beginn verweisen sowohl die Stufe als auch das Canary auf die gleiche API-Version. Wir nutzen die Begriffe Stufen- und Produktionsrelease und Canary und Canary-Release in diesem Abschnitt gleichwertig.

Um eine API mit einem Canary-Release bereitzustellen, erstellen Sie eine Canary-Release-Bereitstellung, indem Sie die [Canary-Einstellungen](#) zur [Stufe](#) einer regulären [Bereitstellung](#) hinzufügen. Die Canary-Einstellungen beschreiben das zugrunde liegende Canary-Release und die Stufe repräsentiert die Produktionsfreigabe der API innerhalb dieser Bereitstellung. Um Canary-Einstellungen hinzuzufügen, legen Sie `canarySettings` in der Bereitstellungsstufe fest und geben Sie Folgendes an:

- Eine Bereitstellungs-ID. Diese ID ist zunächst mit der ID der Basisversion, die für die Stufe festgelegt wurde, identisch.
- Ein [Prozentsatz des API-Verkehrs](#) zwischen 0,0 und 100,0 (inklusive) für das Canary-Release.
- [Stufenvariablen für das Canary-Release](#), die die Stufenvariablen des Produktionsreleases überschreiben können.
- Die [Verwendung des Stage-Caches](#) für Canary-Anfragen, sofern der [useStageCache](#) festgelegt und das API-Caching auf der Bühne aktiviert ist.

Nachdem ein Canary-Release aktiviert wurde, kann die Bereitstellungsstufe nicht mit einem anderen nicht-Canary-Release verbunden werden, bis das Canary-Release deaktiviert und die Canary-Einstellungen von der Stufe entfernt wurden.

Wenn Sie die Protokollierung der API-Ausführung aktivieren, werden für das Canary-Release eigenen Protokolle und Metriken für alle Canary-Anforderungen generiert. Sie werden sowohl an eine CloudWatch Log-Protokollgruppe für die Produktionsphase als auch an eine Logs-Protokollgruppe für die Kanaren CloudWatch gemeldet. Gleiches gilt für die Zugriffsprotokollierung. Die separaten Canary-spezifischen Protokolle sind hilfreich, um neue API-Änderungen zu validieren und zu entscheiden, ob die Änderungen akzeptiert werden sollen und das Canary-Release in die Produktionsstufe übernommen werden soll oder ob die Änderungen verworfen werden sollen und das Canary-Release aus der Produktionsstufe zurückgenommen werden soll.

Die Protokollgruppe für die Ausführung in der Produktionsstufe heißt `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` und die Protokollgruppe für die Ausführung im Canary-Release heißt `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Für die Zugriffsprotokollierung müssen Sie eine neue Protokollgruppe anlegen oder eine bestehende auswählen. Der Name der Canary-Release-Zugriffsprotokollgruppe enthält das Suffix `/Canary`, das an den Namen der ausgewählten Protokollgruppe angehängt ist.

Eine Canary-Version kann den Stage-Cache verwenden, falls dieser aktiviert ist, um Antworten zu speichern und zwischengespeicherte Einträge zu verwenden, um Ergebnisse innerhalb eines vorkonfigurierten Zeitraums `time-to-live (TTL)` an die nächsten Canary-Anfragen zurückzugeben.

Bei einer Canary-Release-Bereitstellung können das Produktions- und das Canary-Release der API der gleichen Version oder verschiedenen Versionen zugeordnet werden. Wenn sie verschiedenen Versionen zugeordnet sind, werden die Antworten für Produktions- und Canary-Anforderungen getrennt zwischengespeichert und der Stage-Cache liefert entsprechende Ergebnisse für Produktions- und Canary-Anforderungen. Wenn das Produktionsrelease und das Canary-Release mit derselben Bereitstellung verbunden sind, verwendet der Stage-Cache einen einzigen Cache-Key für beide Arten von Anforderungen und gibt dieselbe Antwort für dieselben Anforderungen aus dem Produktionsrelease und dem Canary-Release zurück.

Erstellen einer Canary-Release-Bereitstellung

Sie erstellen eine Canary-Release-Bereitstellung bei der Bereitstellung der API mit [Canary-Einstellungen](#) als zusätzliche Eingabe für die [Bereitstellungserstellung](#).

Sie können auch aus einer bestehenden Nicht-Canary-Bereitstellung eine Canary-Release-Bereitstellung erstellen, indem Sie eine [stage:update](#)-Anforderung erstellen, um die Canary-Einstellungen für die Stufe hinzuzufügen.

Beim Erstellen einer nicht-Canary-Release-Bereitstellung können Sie einen nicht existierenden Stufennamen angeben. API Gateway erstellt eine Stufe, wenn die angegebene Stufe nicht existiert. Sie können jedoch beim Anlegen einer Canary-Release-Bereitstellung keinen nicht existierenden Stufennamen angeben. Sie erhalten eine Fehlermeldung und API Gateway erstellt keine Canary-Release-Bereitstellung.

Sie können eine Canary-Release-Bereitstellung in API Gateway mit der API Gateway Gateway-Konsole AWS CLI, dem oder einem AWS SDK erstellen.

Themen

- [Erstellen einer Canary-Bereitstellung mithilfe der API Gateway-Konsole](#)

- [Erstellen Sie eine Canary-Bereitstellung mit dem AWS CLI](#)

Erstellen einer Canary-Bereitstellung mithilfe der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole zu erstellen:

So erstellen Sie eine erste Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene REST API aus oder erstellen Sie eine neue REST API.
3. Klicken Sie im Hauptnavigationsbereich von auf Resources (Ressourcen) und auf Deploy API (API bereitstellen). Befolgen Sie die Anweisungen auf der Seite in Deploy API, um die API in einer neuen Stufe bereitzustellen.

Bislang haben Sie die API in einer Produktions-Release-Stufe bereitgestellt. Als Nächstes konfigurieren Sie Canary-Einstellungen für die Stufe und, falls erforderlich, auch das Caching. Sie legen Stufenvariablen fest oder konfigurieren die API-Ausführung oder die Zugriffsprotokolle.

4. Um das API-Caching zu aktivieren oder der Phase eine AWS WAF Web-ACL zuzuordnen, wählen Sie im Abschnitt Stage-Details die Option Bearbeiten aus. Für weitere Informationen siehe [the section called “Cache-Einstellungen”](#) oder [the section called “So verknüpfen Sie mithilfe der API Gateway Gateway-Konsole eine AWS WAF Web-ACL mit einer API-Gateway-API-Stufe”](#).
5. Um die Ausführung zu konfigurieren oder auf die Protokollierung zuzugreifen, wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) und folgen Sie den Anweisungen am Bildschirm. Weitere Informationen finden Sie unter [Richten Sie die CloudWatch Protokollierung für REST APIs in API Gateway ein](#).
6. Um Stufenvariablen festzulegen, wählen Sie die Registerkarte Stage variables (Stufenvariablen) und folgen Sie den Anweisungen am Bildschirm, um Stufenvariablen hinzuzufügen oder zu ändern. Weitere Informationen finden Sie unter [the section called “Verwenden von Stufenvariablen”](#).
7. Wählen Sie die Registerkarte Canary und dann Create canary (Canary erstellen) aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Canary anzuzeigen.
8. Geben Sie unter Canary settings (Canary-Einstellungen) für Canary den Prozentsatz der Anfragen ein, die an den Canary weitergeleitet werden sollen.

9. Wählen Sie bei Bedarf Stage Cache (Stufen-Cache) aus, um das Caching für die Canary-Version zu aktivieren. Der Cache steht für das Canary-Release erst dann zur Verfügung, wenn das API-Caching aktiviert ist.
10. Um bestehende Stufenvariablen zu überschreiben, geben Sie für Canary Override (Überschreibung des Canary) einen neuen Wert für die Stufenvariable ein.

Nachdem das Canary-Release in der Bereitstellungsstufe initialisiert wurde, ändern Sie die API und testen die Änderungen. Sie können die API in derselben Stufe erneut bereitstellen, sodass sowohl die aktualisierte Version als auch die Basisversion über dieselbe Stufe zugreifbar sind. In den folgenden Schritten wird beschrieben, wie Sie dies tun.

So stellen Sie die neueste API-Version für ein Canary bereit

1. Wählen Sie bei jeder Aktualisierung der API Deploy API (API bereitstellen) aus.
2. Wählen Sie in Deploy API (API bereitstellen) in der Dropdown-Liste Deployment stage (Bereitstellungsstufe) die Stufe aus, die einen Canary enthält.
3. (Optional) Geben Sie unter Deployment description (Beschreibung der Bereitstellen) eine Beschreibung ein.
4. Wählen Sie Deploy, um die neueste API-Version auf das Canary-Release zu übertragen.
5. Falls erwünscht, rekonfigurieren Sie die Stufeneinstellungen, Protokolle oder Canary-Einstellungen, wie in beschriebene [So erstellen Sie eine erste Canary-Release-Bereitstellung](#).

Dadurch verweist das Canary-Release auf die neueste Version, während das Produktionsrelease noch auf die erste Version der API verweist. [canarySettings](#) hat jetzt einen neuen deploymentId-Wert, während die Stufe noch den ursprünglichen [deploymentId](#)-Wert aufweist. Hinter den Kulissen ruft die Konsole [stage:update](#) auf.

Erstellen Sie eine Canary-Bereitstellung mit dem AWS CLI

So erstellen Sie eine Canary-Bereitstellung für eine neue Stufe

1. Verwenden Sie den folgenden [create-deployment](#)-Befehl, um eine Bereitstellung mit zwei Stufenvariablen, aber ohne Canary zu erstellen:

```
aws apigateway create-deployment \  
  --variables sv0=val0,sv1=val1 \  
  --rest-api-id abcd1234 \  

```

```
--stage-name 'prod'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "id": "du4ot1",
  "createdDate": 1511379050
}
```

2. Verwenden Sie den folgenden [create-deployment](#)-Befehl, um eine Canary-Bereitstellung in der prod-Stufe zu erstellen:

```
aws apigateway create-deployment \
  --rest-api-id abcd1234 \
  --canary-settings
percentTraffic=10.5,stageVariableOverrides={sv1='val2',sv2='val3'},useStageCache=false
\
  --stage-name 'prod'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "id": "a6rox0",
  "createdDate": 1511379433
}
```

Die resultierende Bereitstellungs-id identifiziert die Testversion der API für das Canary-Release. Dadurch ist die zugehörige Stufe für Canary aktiviert.

3. (Optional) Verwenden Sie den folgenden [get-stage](#)-Befehl, um die Repräsentation der Stufe anzuzeigen:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Nachfolgend ist eine Repräsentation der Stage als Ausgabe des Befehls zu sehen:

```
{
  "stageName": "prod",
  "variables": {
    "sv0": "val0",
    "sv1": "val1"
  }
}
```



```

    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "du4ot1",
    "lastUpdatedDate": 1511379433,
    "createdDate": 1511379050,
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "a6rox0",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}

```

In diesem Beispiel verwendet die Basisversion der API die Stufenvariablen {"sv0":val0", "sv1":val1"}, während die Testversion die Stufenvariablen {"sv1":val2", "sv2":val3"} verwendet. Sowohl das Produktionsrelease als auch das Canary-Release verwenden die gleiche Stufenvariable sv1, jedoch mit unterschiedlichen Werten (val1 und val2). Die Stufenvariable sv0 wird ausschließlich im Produktionsrelease verwendet und die Stufenvariable sv2 wird nur im Canary-Release verwendet.

Sie können auch eine Canary-Release-Bereitstellung aus einer bestehenden regulären Bereitstellung erstellen, indem Sie die Stufe aktualisieren, um einen Canary zu aktivieren.

So erstellen Sie eine Canary-Release-Bereitstellung aus einer vorhandenen Bereitstellung

1. Verwenden Sie den [create-deployment](#)-Befehl, um eine Bereitstellung ohne einen Canary zu erstellen:

```

aws apigateway create-deployment \
  --variables sv0=val0,sv1=val1 \
  --rest-api-id abcd1234 \
  --stage-name 'beta'

```

2. Verwenden Sie den [update-stage](#)-Befehl, um die Stufe zu aktualisieren und einen Canary zu aktivieren:

```
aws apigateway update-stage \  
  --rest-api-id abcd1234 \  
  --stage-name 'beta' \  
  --patch-operations ' [{  
    "op": "replace",  
    "value": "0.0",  
    "path": "/canarySettings/percentTraffic"  
  }, {  
    "op": "copy",  
    "from": "/canarySettings/stageVariable0overrides",  
    "path": "/variables"  
  }, {  
    "op": "copy",  
    "from": "/canarySettings/deploymentId",  
    "path": "/deploymentId"  
  } ] '
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "stageName": "beta",  
  "variables": {  
    "sv0": "val0",  
    "sv1": "val1"  
  },  
  "cacheClusterEnabled": false,  
  "cacheClusterStatus": "NOT_AVAILABLE",  
  "deploymentId": "cifeiw",  
  "lastUpdatedDate": 1511381930,  
  "createdDate": 1511380879,  
  "canarySettings": {  
    "percentTraffic": 10.5,  
    "deploymentId": "cifeiw",  
    "useStageCache": false,  
    "stageVariable0overrides": {  
      "sv2": "val3",  
      "sv1": "val2"  
    }  
  },  
  "methodSettings": {}  
}
```

Da Sie einen Canary in einer bestehenden Version der API aktiviert haben, verweisen sowohl die Produktionsveröffentlichung (Stage) als auch die Canary-Veröffentlichung (`canarySettings`) auf dieselbe Bereitstellung. Nachdem Sie die API geändert und für diese Stufe wieder bereitgestellt haben, ist die neue Version im Canary-Release. Die Basisversion verbleibt im Produktions-Release. Dies wirkt sich auf die Stufenentwicklung aus, wenn die `deploymentId` im Canary-Release auf die neue Bereitstellungs-id aktualisiert wird und die `deploymentId` im Produktions-Release unverändert bleibt.

Aktualisieren eines Canary-Releases

Nach der Bereitstellung eines Canary-Releases können Sie den prozentualen Anteil des Canary-Verkehrs anpassen oder die Verwendung eines Stage-Caches aktivieren oder deaktivieren, um die Testperformance zu optimieren. Sie können außerdem Stufenvariablen modifizieren, die im Canary-Release verwendet werden, wenn der Ausführungskontext aktualisiert wird. Um solche Aktualisierungen vorzunehmen, rufen Sie die Operation [stage:update](#) mit neuen Werten für [canarySettings](#) auf.

Sie können eine Canary-Version mit der API Gateway Gateway-Konsole, dem Befehl AWS CLI [update-stage](#) oder einem AWS SDK aktualisieren.

Themen

- [Aktualisieren eines Canary-Releases mit der API Gateway-Konsole](#)
- [Aktualisieren Sie eine Canary-Version mit dem AWS CLI](#)

Aktualisieren eines Canary-Releases mit der API Gateway-Konsole

Um die API Gateway-Konsole zur Aktualisierung der vorhandenen Canary-Einstellungen einer Stufe zu nutzen, führen Sie die folgenden Schritte aus:

Aktualisieren bestehender Canary-Einstellungen

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende REST API aus.
2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary und anschließend Edit (Bearbeiten) aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Canary anzuzeigen.

4. Aktualisieren Sie Request distribution (Verteilung anfordern), indem Sie den Prozentsatz auf einen Wert zwischen 0,0 und 100,0 (inklusive) erhöhen oder verringern.
5. Aktivieren oder deaktivieren Sie das Kontrollkästchen unter Stage cache (Stage-Cache).
6. Fügen Sie Canary stage variables (Canary-Stufenvariablen) hinzu, entfernen oder ändern Sie sie.
7. Wählen Sie Speichern.

Aktualisieren Sie eine Canary-Version mit dem AWS CLI

Um einen Canary AWS CLI zu aktualisieren, verwenden Sie den [update-stage](#) Befehl und ändern Sie den Patch-Vorgang für jeden Parameter des Canary.

Der folgende [update-stage](#)-Befehl aktualisiert, ob der Canary den Stufen-Cache verwendet:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Der folgende [update-stage](#)-Befehl aktualisiert den Prozentsatz des Canary-Datenverkehrs:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Der folgende [update-stage](#)-Befehl aktualisiert Stufenvariablen. Das Beispiel zeigt, wie Sie eine neue Stufenvariable mit dem Namen newVar erstellen, die Stufenvariable var2 überschreiben und die Stufenvariable var1 entfernen:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations '[{  
    "op": "replace",  
    "path": "/canarySettings/stageVariable0overrides/newVar",  
    "value": "newVal"  
  }, {
```

```

    "op": "replace",
    "path": "/canarySettings/stageVariable0overrides/var2",
    "value": "val4"
  }, {
    "op": "remove",
    "path": "/canarySettings/stageVariable0overrides/var1"
  }]

```

Sie können das oben Genannte aktualisieren, indem Sie die Operationen zu einem einzigen patch-operations-Wert zusammenfassen:

```

aws apigateway update-stage \
  --rest-api-id {rest-api-id} \
  --stage-name '{stage-name}' \
  --patch-operations ' [{
    "op": "replace",
    "path": "/canarySettings/percentTraffic",
    "value": "20.0"
  }, {
    "op": "replace",
    "path": "/canarySettings/useStageCache",
    "value": "true"
  }, {
    "op": "remove",
    "path": "/canarySettings/stageVariable0overrides/var1"
  }, {
    "op": "replace",
    "path": "/canarySettings/stageVariable0overrides/newVar",
    "value": "newVal"
  }, {
    "op": "replace",
    "path": "/canarySettings/stageVariable0overrides/val2",
    "value": "val4"
  }]

```

Hochstufen eines Canary-Releases

Wenn Sie ein Canary-Release hochstufen, ersetzt das Canary-Release die aktuellen Einstellungen der Stufe. Das Hochstufen eines Canary-Releases bedeutet nicht, dass das Canary für die Stufe deaktiviert wird. Um ein Canary zu deaktivieren, müssen Sie die Canary-Einstellungen der Stufe entfernen. Sie stufen ein Canary-Release wie folgt hoch.

- Setzen Sie die [Bereitstellungs-ID](#) der Stufe mit den [Bereitstellungs-ID](#)-Einstellungen des Canarys zurück. Dadurch wird der API-Snapshot der Stufe mit dem Snapshot des Canarys aktualisiert, sodass die Testversion auch zum Produktionsrelease wird.
- Aktualisieren Sie Stufenvariablen mit Canary-Stufenvariablen, falls vorhanden. Dadurch wird der API-Ausführungskontext der Stufe mit dem des Canarys aktualisiert. Ohne dieses Update kann die neue API-Version unerwartete Ergebnisse liefern, wenn die Testversion unterschiedliche Stufenvariablen oder unterschiedliche Werte vorhandener Stufenvariablen verwendet.
- Legen Sie den Prozentsatz des Canary-Verkehrs auf 0,0 % fest.

Themen

- [Hochstufen eines Canary-Releases mit der API Gateway-Konsole](#)
- [Bewerben Sie eine Canary-Version mit dem AWS CLI](#)

Hochstufen eines Canary-Releases mit der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole hochzustufen:

Hochstufen einer Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende API im Hauptnavigationsbereich aus.
2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary.
4. Wählen Sie Promote Canary (Canary hochstufen) aus.
5. Bestätigen Sie die durchzuführenden Änderungen und wählen Sie Promote canary (Canary hochstufen) aus.

Nach der Aktion verweist das Produktionsrelease auf die gleiche API-Version (deploymentId) wie das Canary-Release. Sie können dies mit dem überprüfen AWS CLI. Ein Beispiel finden Sie unter [the section called “Bewerben Sie eine Canary-Version mit dem AWS CLI”](#).

Bewerben Sie eine Canary-Version mit dem AWS CLI

Um eine Canary-Version mithilfe der AWS CLI Befehle zur Produktionsversion hochzustufen, rufen Sie den `update-stage` Befehl auf, um die Canary-assoziierte `deploymentId` Version in die Stage-Associated zu kopierendeploymentId, den Prozentsatz des Canary-Traffics auf Null (0.0) zurückzusetzen und alle kanariengebundenen Stage-Variablen in die entsprechenden Stage-Variablen zu kopieren.

Angenommen, wir haben eine Canary-Release-Bereitstellung, die durch eine in etwa wie folgt aussehende Stufe beschrieben wird:

```
{
  "_links": {
    ...
  },
  "accessLogSettings": {
    ...
  },
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "canarySettings": {
    "deploymentId": "eh1sby",
    "useStageCache": false,
    "stageVariableOverrides": {
      "sv2": "val3",
      "sv1": "val2"
    },
    "percentTraffic": 10.5
  },
  "createdDate": "2017-11-20T04:42:19Z",
  "deploymentId": "nfc0x",
  "lastUpdatedDate": "2017-11-22T00:54:28Z",
  "methodSettings": {
    ...
  },
  "stageName": "prod",
  "variables": {
    "sv1": "val1"
  }
}
```

Verwenden Sie den folgenden [update-stage](#)-Befehl, um den Canary hochzustufen:

```
aws apigateway update-stage \
  --rest-api-id {rest-api-id} \
  --stage-name '{stage-name}' \
  --patch-operations ' [{
    "op": "replace",
    "value": "0.0",
    "path": "/canarySettings/percentTraffic"
  }, {
    "op": "copy",
    "from": "/canarySettings/stageVariableOverrides",
    "path": "/variables"
  }, {
    "op": "copy",
    "from": "/canarySettings/deploymentId",
    "path": "/deploymentId"
  } ]'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "_links": {
    ...
  },
  "accessLogSettings": {
    ...
  },
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "canarySettings": {
    "deploymentId": "eh1sby",
    "useStageCache": false,
    "stageVariableOverrides": {
      "sv2": "val3",
      "sv1": "val2"
    },
    "percentTraffic": 0
  },
  "createdDate": "2017-11-20T04:42:19Z",
  "deploymentId": "eh1sby",
  "lastUpdatedDate": "2017-11-22T05:29:47Z",
  "methodSettings": {
    ...
  },
}
```



```
"stageName": "prod",
"variables": {
  "sv2": "val3",
  "sv1": "val2"
}
}
```

Durch ein Hochstufen eines Canary-Release wird Canary nicht deaktiviert und die Bereitstellung bleibt weiterhin eine Canary-Release-Bereitstellung. Um daraus eine reguläre Produktions-Release-Bereitstellung zu machen, müssen Sie die Canary-Einstellungen deaktivieren. Weitere Informationen dazu, wie Sie eine Canary-Release-Bereitstellung deaktivieren, finden Sie unter [the section called “Deaktivieren eines Canary-Releases”](#).

Deaktivieren eines Canary-Releases

Um eine Canary-Release-Bereitstellung zu deaktivieren, müssen Sie [canarySettings](#) auf null setzen, um es von der Stufe zu entfernen.

Sie können eine Canary-Release-Bereitstellung mit der API Gateway Gateway-Konsole AWS CLI, dem oder einem AWS SDK deaktivieren.

Themen

- [Deaktivieren eines Canary-Releases mit der API Gateway-Konsole](#)
- [Schalten Sie eine Canary-Version mit dem aus AWS CLI](#)

Deaktivieren eines Canary-Releases mit der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole zu deaktivieren:

Deaktivieren einer Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende API im Hauptnavigationsbereich aus.
2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary.
4. Wählen Sie Löschen aus.
5. Bestätigen Sie, dass Sie das Canary löschen möchten, indem Sie Delete auswählen.

Dadurch wird die Eigenschaft [canarySettings](#) auf null gesetzt und aus der [Bereitstellungsstufe](#) entfernt. Sie können dies mit dem überprüfen AWS CLI. Ein Beispiel finden Sie unter [the section called “Schalten Sie eine Canary-Version mit dem aus AWS CLI”](#).

Schalten Sie eine Canary-Version mit dem aus AWS CLI

Der folgende [update-stage](#)-Befehl deaktiviert die Bereitstellung der Canary-Version:

```
aws apigateway update-stage \  
  --rest-api-id abcd1234 \  
  --stage-name canary \  
  --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

Die Ausgabe sollte wie folgt aussehen:

```
{  
  "stageName": "prod",  
  "accessLogSettings": {  
    ...  
  },  
  "cacheClusterEnabled": false,  
  "cacheClusterStatus": "NOT_AVAILABLE",  
  "deploymentId": "nfcn0x",  
  "lastUpdatedDate": 1511309280,  
  "createdDate": 1511152939,  
  "methodSettings": {  
    ...  
  }  
}
```

Wie aus der Ausgabe hervorgeht, ist die Eigenschaft [canarySettings](#) nun nicht mehr in der [Stufe](#) einer Canary-deaktivierten Bereitstellung enthalten.

REST-Updates APIs , die eine erneute Bereitstellung erfordern

Die Pflege einer API ist gleichzusetzen mit der Anzeige, Aktualisierung und Löschung der vorhandenen API-Einrichtungen. Sie können eine API mithilfe der API Gateway Gateway-Konsole,, AWS CLI CloudFormation, eines SDK oder der API Gateway Gateway-REST-API verwalten. Die Aktualisierung einer API umfasst die Änderung bestimmter Ressourceneigenschaften oder Konfigurationseinstellungen der API. Während Ressourcenaktualisierungen die erneute Bereitstellung der API erfordern, ist die für Konfigurationsaktualisierungen nicht erforderlich.

In der folgenden Tabelle werden API-Konfigurationen beschrieben, die bei einer Aktualisierung eine erneute Bereitstellung Ihrer API erfordern.

Ressource	Hinweise
ApiKey	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter apikey:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
Authorizer	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter authorizer:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
disableExecuteApiEndpoint	Die Aktualisierung erfordert das Ändern einer beliebigen Stufe der API, zum Beispiel das erneute Bereitstellen der API in einer Stufe.
DocumentationPart	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter documentationpart:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
DocumentationVersion	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter documentationversion:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
GatewayResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter gatewayresponse:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
Integration	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter integration:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
IntegrationResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter integrationresponse:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.

Ressource	Hinweise
Methode	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter method:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
MethodResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter methodresponse:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
Model	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter model:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
RequestValidator	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter requestvalidator:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
Ressource	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter resource:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.
RestApi	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter restapi:update . Die Aktualisierung erfordert die erneute Bereitstellung der API. Dazu gehört auch die Änderung der Ressourcennichtlinien.
VpcLink	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter vpclink:update . Die Aktualisierung erfordert die erneute Bereitstellung der API.

In der folgenden Tabelle werden API-Konfigurationen beschrieben, die bei einer Aktualisierung keine erneute Bereitstellung Ihrer API erfordern.

Konfiguration	Hinweise
Account	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter account:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Bereitstellung	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter deployment:update .
DomainName	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter domainname:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.
BasePathMapping	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter basepathmapping:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.
IP-Adresstyp	Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Stage	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter stage:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Usage	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter usage:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.
UsagePlan	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter usageplan:update . Die Aktualisierung erfordert keine erneute Bereitstellung der API.

Benutzerdefinierter Domainname für öffentliches REST APIs in API Gateway

Benutzerdefinierte Domainnamen sind einfacher und intuitiver URLs und können Ihren API-Benutzern zur Verfügung gestellt werden.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

wobei von API Gateway generiert *api-id* wird, die AWS Region *region* ist und von Ihnen bei der Bereitstellung der API angegeben wird.

Der Hostname-Teil der URL (also *api-id*.execute-api.*region*.amazonaws.com) verweist auf einen API-Endpunkt. Der Standardname des API-Endpunkts wird zufällig generiert, ist schwer zu merken und nicht besonders benutzerfreundlich.

Mit benutzerdefinierten Domainnamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. myservice) auswählen, um die alternative URL Ihrer API zuzuweisen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

Note

Informationen zu benutzerdefinierten Domainnamen für private APIs Benutzer finden Sie unter [the section called “Benutzerdefinierte Domainnamen für private APIs”](#).

Überlegungen

Die folgenden Überlegungen können sich auf Ihre Verwendung eines benutzerdefinierten Domainnamens auswirken:

- Sie können den Standardendpunkt Ihrer API deaktivieren. Clients können weiterhin eine Verbindung mit Ihrem Standardendpunkt herstellen, erhalten jedoch einen 403 Forbidden-Statuscode.
- Ein regionaler benutzerdefinierter Domainname kann mit REST APIs und HTTP verknüpft werden APIs. Sie können [API Gateway Version 2](#) verwenden APIs, um regionale benutzerdefinierte Domainnamen für REST zu erstellen und zu verwalten APIs.
- Ein benutzerdefinierter Domainname muss innerhalb einer Region für alle AWS Konten eindeutig sein.

- Sie können Ihren benutzerdefinierten Domainnamen zwischen Edge-optimierten und regionalen Endpunkten migrieren, jedoch kann eine öffentliche benutzerdefinierte Domain nicht zu einem privaten benutzerdefinierten Domainnamen migriert werden.
- Sie müssen den Ressourceneintrag Ihres DNS-Anbieters erstellen oder aktualisieren, bevor Sie ihn dem API-Endpunkt zuweisen können. Ohne diese Zuweisung können API-Anfragen, die an den benutzerdefinierten Domainnamen gerichtet sind, API Gateway nicht erreichen.
- Mit einem Wildcard-Zertifikat können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das Standardkontingent zu überschreiten. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen mit Platzhalter”](#).
- Sie können eine Sicherheitsrichtlinie für Ihren benutzerdefinierten Domainnamen auswählen. Weitere Informationen finden Sie unter [the section called “Eine Sicherheitsrichtlinie auswählen”](#).
- Sie müssen einen regionalen benutzerdefinierten Domainnamen verwenden und die Sicherheitsrichtlinie TLS 1.2 anwenden, um API-Zuweisungen mit mehreren Ebenen zu konfigurieren.

Voraussetzungen für benutzerdefinierte Domainnamen

Die folgenden Voraussetzungen gelten für die Erstellung eines öffentlichen oder privaten benutzerdefinierten Domainnamens. Informationen zu benutzerdefinierten Domainnamen für private APIs Zwecke finden Sie unter [the section called “Benutzerdefinierte Domainnamen für private APIs”](#).

Registrieren eines Domainnamens

Sie müssen über einen registrierten Internet-Domainnamen verfügen, um benutzerdefinierte Domainnamen für Ihre einrichten zu können APIs. Sie können Ihren Internet-Domainnamen über [Amazon Route 53](#) oder über einen Domain-Registrar Ihrer Wahl registrieren. Ihr benutzerdefinierter Domainname kann der Name einer Subdomain oder Root-Domain (Zonen-Apex) einer registrierten Internetdomain sein.

Ihr Domainname muss der [RFC 1035](#)-Spezifikation entsprechen und darf maximal 63 Oktette pro Label und insgesamt 255 Oktette umfassen.

Angaben des Zertifikats für benutzerdefinierte Domainnamen

Bevor Sie einen benutzerdefinierten Domainnamen für eine API einrichten, müssen Sie ein SSL/TLS Zertifikat in ACM bereithalten. Wenn ACM in der AWS Region, in der Sie Ihren benutzerdefinierten Domainnamen erstellen, nicht verfügbar ist, müssen Sie in dieser Region ein Zertifikat in API Gateway importieren.

Um ein SSL/TLS Zertifikat zu importieren, müssen Sie die Zertifizierungsstelle im PEM-Format SSL/TLS, ihren privaten Schlüssel und die Zertifikatskette für den benutzerdefinierten Domainnamen angeben.

Ein in ACM gespeichertes Zertifikat wird durch seinen ARN identifiziert. Mit von ACM ausgestellten Zertifikaten müssen Sie sich keine Sorgen machen, dass vertrauliche Zertifikatsdaten wie der private Schlüssel offengelegt werden. Um ein AWS verwaltetes Zertifikat für einen Domainnamen zu verwenden, verweisen Sie einfach auf dessen ARN.

Wenn Ihre Anwendung Zertifikat-Pinning, manchmal auch SSL-Pinning genannt, verwendet, um ein ACM-Zertifikat anzuheften, kann die Anwendung nach der AWS Verlängerung des Zertifikats möglicherweise keine Verbindung zu Ihrer Domain herstellen. Weitere Informationen finden Sie unter [Probleme beim Zertifikats-Pinning](#) im AWS Certificate Manager -Benutzerhandbuch.

Benutzerdefinierte Domainnamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domainnamen können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das [Standardkontingent](#) zu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domainnamen geben, *customername*.example.com.

Um einen benutzerdefinierten Platzhalter-Domainnamen zu erstellen, geben Sie einen Platzhalter (*) als erste Subdomain eines benutzerdefinierten Domainnamen an, der alle möglichen Subdomains einer Root-Domain darstellt.

Der benutzerdefinierte Platzhalter-Domainname *.example.com erzeugt beispielsweise Subdomains wie a.example.com, b.example.com und c.example.com. Wenn Sie den benutzerdefinierten Platzhalter-Domainnamen erstellen, werden alle seine Subdomains entsprechend dem Routing-Modus des Platzhalter-Domainnamens weitergeleitet. Um Subdomains an andere weiterzuleiten APIs, können Sie einen der folgenden Schritte ausführen:

- Verwenden Sie Routing-Regeln, um eingehende Anfragen APIs mithilfe des Host Headers *.example.com an ein anderes REST-Ziel weiterzuleiten. Weitere Informationen finden Sie unter [the section called “Beispiel 4: Routing-Regeln für Platzhalter-Domainnamen”](#).
- Einen eigenen Domainnamen für Subdomains erstellen, die zu einem anderen Endpunkt geleitet werden sollen. In einem einzigen AWS Konto können Sie *.example.com sowohl als auch habena.example.com.

Sie können die Kontextvariablen `$context.domainName` und `$context.domainPrefix` verwenden, um den Domainnamen zu bestimmen, den ein Client für den Aufruf Ihrer API genutzt hat.

Weitere Informationen zu Kontextvariablen finden Sie unter [Variablen für Datentransformationen für API Gateway](#).

Um einen benutzerdefinierten Platzhalter-Domainnamen zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat vorlegen, das entweder über die DNS- oder die E-Mail-Validierungsmethode validiert wurde.

 Note

Sie können keinen benutzerdefinierten Platzhalter-Domänennamen erstellen, wenn ein anderes AWS Konto einen benutzerdefinierten Domänennamen erstellt hat, der mit dem benutzerdefinierten Platzhalterdomänennamen in Konflikt steht. Wenn Konto A beispielsweise `a.example.com` erstellt hat, kann Konto B den Platzhalter-Domainnamen `*.example.com` nicht erstellen.

Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das [AWS - Support-Center](#) wenden, um eine Ausnahme zu beantragen.

Nächste Schritte für benutzerdefinierte Domainnamen

Die folgenden Schritte können Sie als Nächstes für benutzerdefinierte Domainnamen ausführen:

Nächste Schritte

- Informationen zum Einrichten Ihres SSL/TLS Zertifikats finden Sie unter [the section called “Bereiten Sie Zertifikate vor in AWS Certificate Manager”](#)
- Informationen zum Erstellen eines regionalen benutzerdefinierten Domainnamens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).
- Informationen zum Erstellen eines Edge-optimierten benutzerdefinierten Domainnamens finden Sie unter [the section called “Einrichten eines Edge-optimierten benutzerdefinierten Domainnamens in API Gateway”](#).
- Informationen zur Migration zwischen regionalen und Edge-optimierten benutzerdefinierten Domainnamen finden Sie unter [the section called “Migrieren von benutzerdefinierten Domainnamen”](#).
- Informationen zum Verbinden von API-Stufen mit einem benutzerdefinierten Domainnamen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

- Informationen zur Auswahl einer Sicherheitsrichtlinie für Ihren benutzerdefinierten Domainnamen finden Sie unter [the section called “Eine Sicherheitsrichtlinie auswählen”](#).
- Informationen zum Deaktivieren des Standardendpunkts für Ihren benutzerdefinierten Domainnamen finden Sie unter [the section called “Deaktivieren des Standardendpunkts”](#).
- Informationen zum Verwenden von Route-53-Zustandsprüfungen zur Steuerung des DNS-Failovers von einer API in API Gateway finden Sie unter [the section called “DNS-Failover”](#).

Wenn Sie zum ersten Mal einen benutzerdefinierten Domainnamen erstellen, beginnen Sie am besten mit [the section called “Bereiten Sie Zertifikate vor in AWS Certificate Manager”](#), um Ihr Zertifikat anzugeben und fahren dann mit [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#) fort, um einen regionalen benutzerdefinierten Domainnamen zu erstellen.

Bereiten Sie Zertifikate vor in AWS Certificate Manager

Bevor Sie einen benutzerdefinierten Domainnamen für eine API einrichten, muss in AWS Certificate Manager ein SSL/TLS-Zertifikat vorhanden sein. Weitere Informationen finden Sie im [AWS Certificate Manager -Benutzerhandbuch](#).

Überlegungen

Im Folgenden finden Sie Hinweise zu Ihrem SSL/TLS Zertifikat.

- Wenn Sie einen Edge-optimierten benutzerdefinierten Domainnamen erstellen, unterstützt API Gateway Zertifikate CloudFront für benutzerdefinierte Domainnamen. Daher werden die Anforderungen und Einschränkungen eines benutzerdefinierten SSL/TLS Domainnamenzertifikats von bestimmt. [CloudFront](#) Beispielsweise beträgt die maximale Länge des öffentlichen Schlüssels 2048 und die Länge des privaten Schlüssels 1024, 2048 und 4096. Die Länge des öffentlichen Schlüssels wird von der ausgewählten Zertifizierungsstelle bestimmt. Wenden Sie sich an Ihre Zertifizierungsstelle, um Schlüssel zurückzugeben, deren Länge vom Standard abweicht. Weitere Informationen finden Sie unter [Sicherer Zugriff auf Ihre Objekte](#) und [Erstellen von signierten URLs und signierten Cookies](#).
- Wenn Sie einen benutzerdefinierten regionalen Domainnamen erstellen, beträgt die maximale Größe des öffentlichen Schlüssels 2048.
- Wenn Sie ein ACM-Zertifikat im Zusammenhang mit einem regionalen benutzerdefinierten Domainnamen verwenden möchten, müssen Sie das Zertifikat in der Region Ihrer API anfordern oder importieren. Das Zertifikat muss den benutzerdefinierten Domainnamen abdecken.

- Wenn Sie ein ACM-Zertifikat im Zusammenhang mit einem Edge-optimierten benutzerdefinierten Domainnamen verwenden möchten, müssen Sie das Zertifikat in der us-east-1-Region (USA Ost (Nord-Virginia)) anfordern oder importieren.
- Sie müssen einen registrierten Domainnamen haben, z. B. *example.com*. Sie können entweder [Amazon Route 53](#) oder einen anderen akkreditierten Domain-Registrar Ihrer Wahl verwenden. Eine Liste dieser Registrare finden Sie unter [Accredited Registrar Directory](#) (Verzeichnis autorisierter Vergabestellen) auf der ICANN-Website.

Um ein SSL/TLS Zertifikat zu erstellen oder in ACM zu importieren

Die folgenden Verfahren zeigen, wie Sie ein SSL/TLS Zertifikat für einen Domainnamen erstellen oder importieren.

To request a certificate provided by ACM for a domain name

1. Melden Sie sich bei der [AWS Certificate Manager -Konsole](#) an.
2. Wählen Sie Ein privates Zertifikat anfordern aus.
3. Klicken Sie unter Zertifikatstyp auf Ein öffentliches Zertifikat anfordern.
4. Wählen Sie Weiter aus.
5. Geben Sie unter Vollständig qualifizierter Domänenname einen benutzerdefinierten Domainnamen für Ihre API ein, z. B. `api.example.com`.
6. Wählen Sie optional Fügen Sie diesem Zertifikat einen weiteren Namen hinzu.
7. Wählen Sie unter Validierungsmethode eine Methode zur Überprüfung des Domäneigentümers aus.
8. Wählen Sie unter Schlüsselalgorithmus einen Verschlüsselungsalgorithmus aus.
9. Wählen Sie Anfrage aus.
10. Damit eine Anforderung gültig ist, muss der registrierte Besitzer einer Internet-Domain der Anforderung vor der Ausstellung des Zertifikats durch ACM zustimmen. Wenn Sie Route 53 verwenden, um Ihre öffentlichen DNS-Einträge zu verwalten, können Sie Ihre Datensätze direkt über die ACM-Konsole aktualisieren.

To import into ACM a certificate for a domain name

1. Holen Sie sich von einer Zertifizierungsstelle ein PEM-codiertes SSL/TLS Zertifikat für Ihren benutzerdefinierten Domainnamen. Eine unvollständige Liste dieser CAs Zertifikate finden Sie in der Liste der von [Mozilla mitgelieferten Zertifizierungsstellen](#).

- a. Erstellen Sie einen privaten Schlüssel für das Zertifikat und speichern Sie die Ausgabe in einer Datei. Verwenden Sie dazu das [OpenSSL](#)-Toolkit auf der OpenSSL-Website:

```
openssl genrsa -out private-key-file 2048
```

- b. Erstellen Sie mithilfe von OpenSSL eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit dem zuvor erstellten privaten Schlüssel:

```
openssl req -new -sha256 -key private-key-file -out CSR-file
```

- c. Senden Sie die Zertifikatssignierungsanforderung an die Zertifizierungsstelle und speichern Sie das Zertifikat.
 - d. Laden Sie die Zertifikatkette bei der Zertifizierungsstelle herunter.

 Note

Wenn Ihnen der private Schlüssel auf andere Art bereitgestellt wird und er verschlüsselt ist, können Sie ihn mit dem folgenden Befehl entschlüsseln, bevor Sie ihn zur Einrichtung eines benutzerdefinierten Domainnamens an API Gateway senden.

```
openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -nocrypt -out MyDecryptedKey.pem
```

2. Laden Sie das Zertifikat hoch auf AWS Certificate Manager:
 - a. Melden Sie sich an der [AWS Certificate Manager -Konsole](#) an.
 - b. Wählen Sie Ein Zertifikat importieren aus.
 - c. Geben Sie unter Zertifikatstext den Text des PEM-formatierten Serverzertifikats von Ihrer Zertifizierungsstelle ein. Im Folgenden sehen Sie ein verkürztes Beispiel eines solchen Zertifikats.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++g0fQtj0IBoqDANBgqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

- d. Geben Sie unter Privater Schlüssel des Zertifikats den privaten Schlüssel Ihres PEM-formatierten Zertifikats ein. Im Folgenden sehen Sie ein verkürztes Beispiel eines solchen Schlüssels.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nET0
...
1qGvJ3u04vdnzaYN5WoyN5LFckr1A71+CszD1CGSqBVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

- e. Geben Sie unter Zertifikatkette nacheinander und ohne Leerzeilen die PEM-formatierten Zwischenzertifikate und optional das Root-Zertifikat ein. Wenn Sie das Stammzertifikat eingeben, muss die Zertifikatkette mit Zwischenzertifikaten beginnen und mit dem Stammzertifikat enden. Verwenden Sie die von der Zertifizierungsstelle bereitgestellten Zwischenzertifikate. Verwenden Sie keine Zwischenzertifikate außerhalb der Vertrauenskette. Das folgende Beispiel zeigt ein verkürztes Beispiel.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWtYdrB5NogYUx1U9Pamy3DANBgqhkiG9w0BAQUFADCB
...
8/ifB1IK3se2e4/hEfcEejX/arxbx1BJCHBv1EPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Im Folgenden ein weiteres Beispiel.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

- f. Klicken Sie auf Weiter und dann erneut auf Weiter.

Wenn das Zertifikat erfolgreich erstellt oder importiert wurde, notieren Sie seinen ARN. Diesen benötigen Sie für die Einrichtung des benutzerdefinierten Domainnamens.

Einrichten eines regionalen benutzerdefinierten Domainnamens in API Gateway

Verwenden Sie einen benutzerdefinierten Domainnamen, um eine benutzerfreundliche API-Basis-URL zu erstellen. Mit einem regionalen benutzerdefinierten Domainnamen können Sie HTTP- und REST-API-Stufen demselben benutzerdefinierten Domainnamen zuweisen und gegenseitige TLS-Authentifizierung verwenden.

Überlegungen

Im Folgenden finden Sie einige Überlegungen zu Ihrem regionalen benutzerdefinierten Domainnamen:

- Die Bereitstellung eines regionenspezifischen ACM-Zertifikats ist erforderlich. Deses Zertifikat muss sich in derselben Region wie Ihre API befinden. Weitere Informationen zum Erstellen oder Hochladen eines benutzerdefinierten Domainnamen-Zertifikats finden Sie unter [the section called “Bereiten Sie Zertifikate vor in AWS Certificate Manager”](#).
- Wenn Sie einen regionalen benutzerdefinierten Domainnamen mit einem ACM-Zertifikat erstellen, generiert API Gateway eine serviceverknüpfte Rolle in Ihrem Konto. Die serviceverknüpfte Rolle ist erforderlich, um Ihr ACM-Zertifikat Ihrem regionalen Endpunkt anzuhängen. Die Rolle wird benannt `AWSServiceRoleForAPIGateway` und die `APIGatewayServiceRolePolicy` verwaltete Richtlinie wird an sie angehängt. Weitere Informationen zur Verwendung von serviceverknüpften Rollen finden Sie unter [Verwenden von serviceverknüpften Rollen](#).
- Nachdem Sie Ihren regionalen benutzerdefinierten Domainnamen erstellt haben, müssen Sie einen DNS-Datensatz erstellen, der den benutzerdefinierten Domainnamen auf die regionale Domain verweist. Dadurch wird der Datenverkehr, der an den benutzerdefinierten Domainnamen gebunden ist, an den regionalen Hostnamen der API weitergeleitet.

Der DNS-Datensatz kann entweder ein CNAME- oder ein A-Alias-Datensatz sein. Wenn Sie Route 53 als DNS-Anbieter verwenden, erstellen Sie einen A-Alias-Datensatz. Wenn Sie einen Drittanbieter-DNS-Dienst nutzen, verwenden Sie einen CNAME-Datensatz. Wenn Sie einen CNAME-Datensatz nutzen und einen Schnittstellen-VPC-Endpunkt für API Gateway mit aktiviertem privatem DNS für eine private API erstellen, können Sie den benutzerdefinierten Domainnamen innerhalb der VPC, die Ihre private API hostet, nicht auflösen.

Erstellen eines regionalen benutzerdefinierten Domainnamens

Im folgenden Verfahren wird das Erstellen eines regionalen benutzerdefinierten Domainnamens erläutert. Nach Abschluss des Verfahrens erstellen Sie eine Routing-Regel, um die Stufen Ihrer API Ihrem benutzerdefinierten Domainnamen zuzuordnen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie Erstellen aus.
4. Geben Sie für Domänenname einen Domainnamen ein.
5. Wählen Sie unter Routing-Modus die Option Nur Routing-Regeln aus.

In diesem Routing-Modus können Sie nur mithilfe von Routing-Regeln Traffic von Ihrem benutzerdefinierten Domainnamen an Ihren APIs senden. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

6. Wählen Sie unter Minimale TLS-Version eine Version aus.
7. Wählen Sie unter Endpunktconfiguration für API-Endpunkttyp die Option Regional aus.
8. Wählen Sie ein ACM-Zertifikat aus. Das Zertifikat muss sich in derselben Region wie die API befinden.
9. Wählen Sie Erstellen aus.

AWS CLI

Der folgende [create-domain-name](#) Befehl erstellt einen benutzerdefinierten Domainnamen:

```
aws apigatewayv2 create-domain-name \  
  --domain-name 'regional.example.com' \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \  
  --routing-mode ROUTING_RULE_ONLY
```

Die Ausgabe sieht wie folgt aus:

```
{
```

```

    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
      {
        "ApiGatewayDomainName": "d-numh1z56v6.execute-api.us-
west-2.amazonaws.com",
        "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
        "DomainNameStatus": "AVAILABLE",
        "EndpointType": "REGIONAL",
        "HostedZoneId": "Z20JLYMU09EFXC",
        "SecurityPolicy": "TLS_1_2"
      }
    ]
    "RoutingMode": "ROUTING_RULE_ONLY"
  ]
}

```

Der Eigenschaftswert `DomainNameConfigurations` gibt den Hostnamen der regionalen API zurück. Sie müssen einen DNS-Datensatz erstellen, der Ihren benutzerdefinierten Domainnamen auf diesen regionalen Domainnamen verweist. Dadurch wird der an den benutzerdefinierten Domainnamen gebundene Datenverkehr an den Hostnamen dieser regionalen API weitergeleitet.

Erstellen einer Routing-Regel für Ihren regionalen benutzerdefinierten Domainnamen

Nachdem Sie Ihren benutzerdefinierten Domainnamen erstellt haben, konfigurieren Sie, wie der Traffic von Ihrem benutzerdefinierten Domainnamen zu Ihrem APIs weitergeleitet wird. Da Sie den Routing-Modus auf eingestellt haben `ROUTING_RULE_ONLY`, verwenden Sie Routing-Regeln, um eingehende Anfragen an Ihren benutzerdefinierten Domainnamen an Ihren APIs weiterzuleiten.

In diesem Beispiel erstellen Sie eine Catch-All-Regel, die alle eingehenden Anfragen an Ihren benutzerdefinierten Domainnamen an eine Stufe Ihrer API weiterleitet. Sie können auch Routing-Regeln konfigurieren, die auf unterschiedlichen Header- und Pfadbedingungen basieren. Weitere Informationen finden Sie unter [the section called “Routing-Regeln”](#).

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie einen benutzerdefinierten Domainnamen aus.
3. Wählen Sie auf der Registerkarte Routing-Details die Option Routing-Regel hinzufügen aus.

4. Wählen Sie Neue Bedingung hinzufügen aus, um eine neue Bedingung hinzuzufügen.
5. Belassen Sie diese Regel ohne Bedingungen. Dadurch werden alle Anfragen an Ihren benutzerdefinierten Domainnamen an Ihre Ziel-API und Zielstufe weitergeleitet.
6. Verwenden Sie für Aktion das Dropdown-Menü, um Ihre Ziel-API und Ihre Zielstufe auszuwählen.
7. Wählen Sie Weiter aus.
8. Geben Sie im Prioritätsfeld **100** ein.

API Gateway wertet Regeln in der Reihenfolge der Priorität aus, ausgehend vom niedrigsten Wert bis hin zum höchsten Wert. Da es sich um eine Catch-All-Regel handelt, verwenden Sie eine hohe Priorität, damit API Gateway alle zusätzlichen Regeln, die Sie erstellen, zuerst abgleichen kann.

9. Wählen Sie Routing-Regel erstellen aus.

AWS CLI

Der folgende `create-routing-rule`-Befehl erstellt eine Auffang-Routing-Regel:

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'regional.example.com' \
  --priority 100 \
  --conditions \
  --actions '[{
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }]'
```

Sie können den Routing-Modus jederzeit ändern und neue Regeln erstellen. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

Erstellen eines DNS-Datensatzes für Ihren regionalen benutzerdefinierten Domainnamen

Nachdem Sie den benutzerdefinierten Domainnamen und die Basispfadzuweisungen festgelegt haben, erstellen Sie einen DNS-Datensatz, der Ihren benutzerdefinierten Domainnamen auf den neu erstellten regionalen Domainnamen verweist.

AWS-Managementkonsole

Um das zu verwenden AWS-Managementkonsole, folgen Sie der [Route 53-Dokumentation zur Konfiguration von Route 53 für die Weiterleitung von Datenverkehr an API Gateway](#).

AWS CLI

Wenn Sie Ihre DNS-Datensätze so konfigurieren möchten, dass der regionale benutzerdefinierte Domainname seinem Hostnamen in der ID der vorgegebenen gehosteten Zone zugewiesen wird, erstellen Sie zunächst eine JSON-Datei, die die Konfiguration für die Einrichtung eines DNS-Datensatzes für den regionalen Domainnamen enthält.

Die folgende Datei `setup-dns-record.json` zeigt, wie Sie einen DNS-A-Datensatz erstellen, um einen regionalen benutzerdefinierten Domainnamen (`regional.example.com`) seinem regionalen Hostnamen (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) zuzuweisen, der im Rahmen der benutzerdefinierten Domainnamenerstellung bereitgestellt wurde. Die Eigenschaften `DNSName` und `HostedZoneId` von `AliasTarget` können die Werte `regionalDomainName` und `regionalHostedZoneId` des benutzerdefinierten Domainnamens entsprechend annehmen. Sie können die Regional Route 53 Hosted Zone auch IDs in [Amazon API Gateway Endpoints and Quotas](#) abrufen.

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "regional.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
          "HostedZoneId": "Z20JLYMU09EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Im Folgenden [change-resource-record-sets](#) wird ein DNS-Eintrag für Ihren regionalen benutzerdefinierten Domainnamen erstellt:

```
aws route53 change-resource-record-sets \
```

```
--hosted-zone-id Z20JLYMU09EFXC \  
--change-batch file://path/to/your/setup-dns-record.json
```

Ersetzen Sie die `hosted-zone-id` durch die Route-53-Hosted-Zone-ID des DNS-Datensatzes, die in Ihrem Konto eingerichtet wurde. Der `change-batch`-Parameterwert zeigt auf eine JSON-Datei (*setup-dns-record.json*) in einem Ordner (*path/to/your*).

Einrichten eines Edge-optimierten benutzerdefinierten Domainnamens in API Gateway

Wenn Sie einen benutzerdefinierten Domainnamen für eine Edge-optimierte API erstellen, richtet API Gateway eine CloudFront Verteilung und einen DNS-Eintrag ein, um den API-Domainnamen dem CloudFront Distributionsdomännennamen zuzuordnen. Anfragen für die API werden dann über die zugeordnete CloudFront Verteilung an API Gateway weitergeleitet. Diese Zuordnung ist für API-Anfragen vorgesehen, die daran gebunden sind, dass der benutzerdefinierte Domainname über die zugeordnete CloudFront Distribution an API Gateway weitergeleitet wird.

Überlegungen

Die folgenden Punkte sind für Ihren Edge-optimierten benutzerdefinierten Domainnamen zu beachten:

- Um einen Edge-optimierten benutzerdefinierten Domainnamen einzurichten oder sein Zertifikat zu aktualisieren, benötigen Sie die Berechtigung, Distributionen zu aktualisieren. CloudFront

Die folgenden Berechtigungen sind erforderlich, um Distributionen zu aktualisieren: CloudFront JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowCloudFrontUpdateDistribution",  
      "Effect": "Allow",  
      "Action": [  
        "cloudfront:updateDistribution"  
      ],  
      "Resource": [  
        "*"   
      ]  
    }  
  ]  
}
```

```
}  
]
```

- Sie müssen ein Zertifikat für Ihren Edge-optimierten benutzerdefinierten Domainnamen in der us-east-1-Region (USA Ost (Nord-Virginia)) anfordern oder importieren.
- Die von API Gateway erstellte CloudFront Distribution gehört einem regionsspezifischen Konto, das mit API Gateway verbunden ist. Bei der Ablaufverfolgung von Vorgängen zur Erstellung und Aktualisierung einer solchen CloudFront Verteilung müssen Sie diese API Gateway Gateway-Konto-ID verwenden. CloudTrail Weitere Informationen finden Sie unter [Protokollieren Sie die Erstellung eines benutzerdefinierten Domainnamens CloudTrail](#).
- API Gateway unterstützt Edge-optimierte benutzerdefinierte Domainnamen, indem es Server Name Indication (SNI) für die Verteilung nutzt. CloudFront Weitere Informationen zur Verwendung benutzerdefinierter Domainnamen in einer CloudFront Distribution, einschließlich des erforderlichen Zertifikatsformats und der maximalen Länge eines Zertifikatsschlüssels, finden Sie [unter Verwenden alternativer Domainnamen und HTTPS](#) im Amazon CloudFront Developer Guide
- Die Fertigstellung eines Edge-optimierten benutzerdefinierte Domainnamens dauert etwa 40 Minuten.
- Nachdem Sie Ihren Edge-optimierten benutzerdefinierten Domainnamen erstellt haben, müssen Sie einen DNS-Eintrag erstellen, um den benutzerdefinierten Domainnamen dem CloudFront Distributionsnamen zuzuordnen.

Erstellen eines Edge-optimierten benutzerdefinierten Domainnamens

Im folgenden Verfahren wird beschrieben, wie Sie einen benutzerdefinierten Domainnamen für eine API erstellen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie Domainnamen hinzufügen aus.
4. Geben Sie unter Domänenname einen Domainnamen ein.
5. Wählen Sie für den Routing-Modus `_ONLY`. `API_MAPPING`
6. Wählen Sie unter API-Endpunkttyp die Option Edge-optimiert aus.
7. Wählen Sie eine minimale TLS-Version aus.

8. Wählen Sie ein ACM-Zertifikat aus.
9. Klicken Sie auf Domainnamen hinzufügen.

REST API

1. Rufen Sie [domainname:create](#) auf und geben Sie den benutzerdefinierten Domainnamen und den ARN eines in AWS Certificate Manager gespeicherten Zertifikats an.

Der erfolgreiche API-Aufruf gibt eine `201 Created` Antwort zurück, die den Zertifikat-ARN sowie den zugehörigen CloudFront Distributionsnamen in der Nutzlast enthält.

2. Notieren Sie sich den Namen der CloudFront Distributionsdomäne, der in der Ausgabe angezeigt wird. Diesen benötigen Sie im nächsten Schritt, um das A-Datensatz-Aliasziel der benutzerdefinierten Domain in Ihrem DNS festzulegen.

Code-Beispiele für diesen REST-API-Aufruf finden Sie unter [domainname:create](#).

Es dauert etwa 40 Minuten, bis ein Edge-optimierter benutzerdefinierter Domainname fertig ist. In der Konsole wird jedoch sofort der zugehörige CloudFront Distributionsdomänenname in der Form von *distribution-id.cloudfront.net* zusammen mit dem Zertifikat-ARN angezeigt. In der Zwischenzeit können Sie eine Basispfadzuweisung oder eine Routingregel erstellen und dann den DNS-Eintragsalias so konfigurieren, dass der benutzerdefinierte Domainname dem zugehörigen CloudFront Distributionsdomänennamen zugeordnet wird.

Konfigurieren einer Basispfadzuweisung einer API mit einem benutzerdefinierten Domainnamen als Hostname

Da Sie den Routingmodus auf eingestellt haben `API_MAPPING_ONLY`, können Sie die Basispfadzuordnung verwenden, um einen einzelnen benutzerdefinierten Domainnamen als Hostnamen von mehreren APIs zu verwenden. So wird eine API mithilfe der Kombination aus benutzerdefiniertem Domainnamen und dem zugehörigen Basispfad zugänglich gemacht.

Wenn Sie beispielsweise eine API mit dem Namen `PetStore` und eine weitere mit dem Namen `Dogs` in API Gateway erstellt haben und dann einen benutzerdefinierten Domainnamen `api.example.com` einrichten, können Sie die URL der `PetStore`-API als `https://api.example.com` festlegen.

Dadurch wird die `PetStore`-API mit dem Basispfad einer leeren Zeichenfolge verknüpft. Wenn Sie die URL der `PetStore`-API auf `https://api.example.com/PetStore` festlegen, verknüpft dies

die PetStore-API mit dem Basispfad von PetStore. Sie können einen Basispfad MyDogList für die Dogs-API zuweisen. Die URL `https://api.example.com/MyDogList` ist dann die Stamm-URL der Dogs-API.

Es kann nur ein regionaler benutzerdefinierter Domainname für das Konfigurieren von API-Zuweisungen auf mehreren Ebenen verwendet werden. Edge-optimierte benutzerdefinierte Domainnamen werden nicht unterstützt. Weitere Informationen finden Sie unter [the section called “API-Mappings”](#).

Das folgende Verfahren richtet API-Zuweisungen ein, um Pfade von Ihrem benutzerdefinierten Domainnamen zu Ihren API-Stufen zuzuordnen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich der API Gateway-Konsole aus.
3. Wählen Sie einen benutzerdefinierten Domainnamen aus.
4. Wählen Sie API-Zuweisungen konfigurieren aus.
5. Wählen Sie Neue Zuweisung hinzufügen aus.
6. Geben Sie die API, die Stufe und den Pfad (optional) für die Zuweisung an.
7. Wählen Sie Speichern aus.

REST API

Rufen Sie [basepathmapping:create](#) für einen bestimmten benutzerdefinierten Domainnamen auf und geben Sie in den Nutzdaten der Anfrage die Eigenschaften `basePath` und `restApiId` sowie eine Bereitstellungs-stage an.

Ein erfolgreicher API-Aufruf gibt die Antwort `201 Created` zurück.

Codebeispiele des REST-API-Aufrufs finden Sie unter [basepathmapping:create](#).

Wenn Sie mehr Flexibilität bei der Weiterleitung von Datenverkehr zu Ihrem wünschen APIs, können Sie den Routing-Modus auf `ROUTING_RULE_ONLY` oder ändern `ROUTING_RULE_THEN_API_MAPPING` und eine Routing-Regel erstellen. Weitere Informationen

finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

Erstellen eines DNS-Eintrags für Ihren Edge-optimierten benutzerdefinierten Domainnamen

Richten Sie den Alias für den DNS-Datensatz ein, nachdem Sie mit die Erstellung Ihres Edge-optimierten benutzerdefinierten Domainnamens angestoßen haben.

Wir empfehlen Ihnen, Route 53 zu verwenden, um einen A-Record-Alias für Ihren benutzerdefinierten Domainnamen zu erstellen und den Namen der CloudFront Distributionsdomäne als Aliasziel anzugeben. Das bedeutet, dass Route 53 den benutzerdefinierten Domännennamen auch dann weiterleiten kann, wenn es sich um einen Zone Apex handelt. Weitere Informationen finden Sie im Amazon Route 53-Entwicklerhandbuch unter [Wählen zwischen Alias- und Nicht-Alias-Ressourcendatensätzen](#).

Anweisungen für Amazon Route 53 finden Sie unter [Weiterleiten des Datenverkehrs an eine Amazon API Gateway API unter Verwendung Ihres Domain-Namens](#) im Amazon Route 53-Entwicklerhandbuch.

Protokollieren Sie die Erstellung eines benutzerdefinierten Domainnamens CloudTrail

Wenn für die Protokollierung von API-Gateway-Aufrufen, die von Ihrem Konto getätigt wurden, aktiviert CloudTrail ist, protokolliert API Gateway die zugehörigen CloudFront Distributionsupdates, wenn ein benutzerdefinierter Domainname für eine API erstellt oder aktualisiert wird. Diese Protokolle sind in us-east-1 verfügbar. Da diese CloudFront Distributionen Eigentum von API Gateway sind, wird jede dieser gemeldeten CloudFront Distributionen durch eines der folgenden regionsspezifischen API Gateway Gateway-Konten und nicht durch die IDs Konto-ID des API-Besitzers identifiziert.

Region	Konto-ID
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028

Region	Konto-ID
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837

Region	Konto-ID
me-central-1	614065512851

Wechseln eines in ACM importierten Zertifikats

ACM übernimmt automatisch die Verlängerung eigener ausgestellter Zertifikate. Sie müssen keine von ACM ausgestellten Zertifikate für Ihre benutzerdefinierten Domainnamen rotieren. CloudFront kümmert sich in Ihrem Namen darum.

Wenn Sie jedoch ein Zertifikat in ACM importieren und für einen benutzerdefinierten Domänennamen verwenden, müssen Sie das Zertifikat vor seinem Ablauf auswechseln. Dies umfasst das Importieren eines neuen Drittanbieter-Zertifikats für den Domänennamen und das Auswechseln des vorhandenen durch ein neues Zertifikat. Diesen Vorgang müssen Sie wiederholen, wenn das neu importierten Zertifikat abläuft. Alternativ können Sie in ACM die Ausstellung eines neuen Zertifikats für den Domänennamen beauftragen und das vorhandene durch das neue auswechseln. Danach können Sie ACM verlassen und CloudFront die Zertifikatsrotation automatisch für Sie übernehmen. Führen Sie die Schritte unter [the section called “Um ein SSL/TLS Zertifikat zu erstellen oder in ACM zu importieren”](#) aus, um ein neues ACM-Zertifikat zu erstellen oder zu importieren.

Im folgenden Verfahren wird beschrieben, wie ein Zertifikat für einen Domainnamen rotiert wird.

Note

Das Rotieren eines in ACM importierten Zertifikats dauert etwa 40 Minuten.

AWS-Managementkonsole

1. Fordern Sie ein Zertifikat in ACM an oder importieren Sie es.
2. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
3. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich der API Gateway-Konsole aus.
4. Wählen Sie einen Edge-optimierten benutzerdefinierten Domainnamen aus.
5. Klicken Sie unter Endpunktkonfiguration auf Bearbeiten.

6. Wählen Sie unter ACM-Zertifikat ein Zertifikat aus der Dropdown-Liste aus.
7. Klicken Sie auf Änderungen speichern, um mit dem Rotieren des Zertifikats für den benutzerdefinierten Domainnamen zu beginnen.

REST API

Rufen Sie die Aktion [domainname:update](#) unter Angabe des ARN des neuen ACM-Zertifikats für den jeweiligen Domänennamen auf.

AWS CLI

Im Folgenden wird das ACM-Zertifikat für einen Edge-optimierten Domainnamen [update-domain-name](#) aktualisiert.

```
aws apigateway update-domain-name \
  --domain-name edge.example.com \
  --patch-operations "op='replace',path='/certificateArn',value='arn:aws:acm:us-east-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'"
```

Im Folgenden wird das ACM-Zertifikat für einen regionalen Domainnamen [update-domain-name](#) aktualisiert.

```
aws apigateway update-domain-name \
  --domain-name regional.example.com \
  --patch-operations "op='replace',path='/regionalCertificateArn',value='arn:aws:acm:us-east-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'"
```

Aufrufen einer API mit benutzerdefinierten Domainnamen bei Verwendung einer Basispfadzuweisung

Sofern die korrekte URL verwendet wird, macht es keinen Unterschied, ob eine API mit einem benutzerdefinierten oder dem standardmäßigen Domänennamen aufgerufen wird.

In den folgenden Beispielen werden ein Satz von Standard URLs - und entsprechenden benutzerdefinierten Werten URLs aus zwei APIs (udxjefundqf3duz) in einer bestimmten Region (us-east-1) und einem bestimmten benutzerdefinierten Domainnamen (api.example.com) verglichen und gegenübergestellt.

API-ID	Stufe	Standard-URL	Basispfad	Custom URL
udxjef	Prod	https://udxjef.execute-api.us-east-1.amazonaws.com/prod	/petstore	https://api.example.com/Tierhandlung
udxjef	tst	https://udxjef.execute-api.us-east-1.amazonaws.com/test	/petdepot	https://api.example.com/tierdepot
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/bookstore	https://api.example.com/buchhandlung
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/bookstand	https://api.example.com/buchständer

Für mehr Flexibilität bei der Weiterleitung von Datenverkehr zu Ihren APIs können Sie eine Routing-Regel erstellen. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

API Gateway unterstützt benutzerdefinierte Domainnamen für eine API über die [Server Name Indication \(SNI\)](#). Sie können die API mit einem benutzerdefinierten Domänennamen über einen Browser oder eine Client-Bibliothek aufrufen, die SNI unterstützt.

API Gateway erzwingt SNI bei der CloudFront Distribution. Informationen zur CloudFront Verwendung benutzerdefinierter Domainnamen finden Sie unter [Amazon CloudFront Custom SSL](#).

Migrieren eines benutzerdefinierten Domainnamens in einen anderen API-Endpunkt in API Gateway

Sie können Ihren benutzerdefinierten Domänennamen zwischen Edge-optimierten und regionalen Endpunkten migrieren. Sie können einen öffentlichen benutzerdefinierten Domainnamen nicht zu einem privaten benutzerdefinierten Domainnamen migrieren. Zunächst fügen Sie den neuen Endpunktconfigurationstyp zur bestehenden Liste `endpointConfiguration.types` für den benutzerdefinierten Domänennamen hinzu. Danach richten Sie einen DNS-Datensatz so ein, dass der benutzerdefinierte Domänenname auf den neu bereitgestellten Endpunkt verweist. Zum Schluss entfernen Sie den veralteten Endpunkt des benutzerdefinierten Domainnamens.

Überlegungen

Im Folgenden finden Sie Überlegungen für die Migration Ihrer benutzerdefinierten Domain von einem regionalen API-Endpunkt zu einem Edge-optimierten API-Endpunkt:

- Ein Edge-optimierter benutzerdefinierter Domainname erfordert ein von ACM bereitgestelltes Zertifikat aus der `us-east-1`-Region (USA Ost (Nord-Virginia)). Dieses Zertifikat wird auf alle geografischen Standorte verteilt.
- Für einen regionalen benutzerdefinierten Domainnamen ist ein Zertifikat erforderlich, das von ACM in derselben Region bereitgestellt wird, in der die API gehostet wird. Sie können einen Edge-optimierten benutzerdefinierten Domainnamen, der sich außerhalb der `us-east-1`-Region befindet, zu einem regionalen benutzerdefinierten Domainnamen migrieren, indem Sie zunächst ein neues ACM-Zertifikat aus der aktuellen Region der API anfordern.
- Die Migration von einem Edge-optimierten zu einem regionalen benutzerdefinierten Domainnamen kann bis zu 60 Sekunden dauern. Die Migrationszeit hängt auch davon ab, wann Sie Ihre DNS-Datensätze aktualisieren.
- Sie können eine zusätzliche Endpunktconfiguration nur hinzufügen, wenn der Endpunktzugriffsmodus auf eingestellt ist `BASIC`. Sobald Sie zwei Endpunktconfigurationen haben, können Sie den Endpunktzugriffsmodus nicht mehr ändern. Weitere Informationen finden Sie unter [the section called “Endpunktzugriffsmodus”](#).
- Wenn Ihr benutzerdefinierter Domainname eine Sicherheitsrichtlinie verwendet, die damit beginnt `SecurityPolicy_`, dass, wenn Sie einen neuen Endpunktconfigurationstyp hinzufügen, der Endpunktzugriffsmodus für beide Endpunkttypen identisch ist, und Sie müssen eine Sicherheitsrichtlinie auswählen, die mit `SecurityPolicy_` für den neuen Endpunktconfigurationstyp beginnt.

Migrieren von benutzerdefinierten Domainnamen

Note

Um die Migration abzuschließen, stellen Sie sicher, dass Sie den veralteten Endpunkt aus Ihrem benutzerdefinierten Domainnamen entfernen.

Gehen Sie wie folgt vor, um einen regionalen benutzerdefinierten Domainnamen zu einem Edge-optimierten benutzerdefinierten Domainnamen zu migrieren.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen Edge-optimierten benutzerdefinierten Domainnamen aus.
4. Klicken Sie unter Endpunktkonfiguration auf Bearbeiten.
5. Wählen Sie Regionalen Endpunkt hinzufügen aus.
6. Wählen Sie unter ACM-Zertifikat ein Zertifikat aus.

Das regionale Zertifikat muss sich in derselben Region befinden wie die regionale API.

7. Wählen Sie Änderungen speichern aus.
8. Richten Sie einen DNS-Datensatz so ein, dass der regionale benutzerdefinierte Domainname auf diesen regionalen Hostnamen verweist. Weitere Informationen finden Sie unter [Konfiguration von Route 53 zur Weiterleitung von Datenverkehr an API Gateway](#).
9. Nachdem Sie bestätigt haben, dass Ihre DNS-Konfiguration den richtigen Endpunkt verwendet, löschen Sie die Edge-optimierte Endpunktkonfiguration. Wählen Sie Ihren benutzerdefinierten Domainnamen aus und klicken Sie dann unter Edge-optimierte Endpunktkonfiguration auf Löschen.
10. Bestätigen Sie Ihre Auswahl und löschen Sie den Endpunkt.

AWS CLI

Der folgende [update-domain-name](#) Befehl migriert einen Edge-optimierten benutzerdefinierten Domainnamen zu einem regionalen benutzerdefinierten Domainnamen:

```
aws apigateway update-domain-name \  
  --domain-name 'api.example.com' \  
  --patch-operations '[  
    { "op": "add", "path": "/endpointConfiguration/types", "value": "REGIONAL" },  
    { "op": "add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-  
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }  
  ]'
```

Das regionale Zertifikat muss sich in derselben Region befinden wie die regionale API.

Die Ausgabe sieht wie folgt aus:

```
{  
  "certificateArn": "arn:aws:acm:us-  
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",  
  "certificateName": "edge-cert",  
  "certificateUploadDate": "2017-10-16T23:22:57Z",  
  "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",  
  "domainName": "api.example.com",  
  "endpointConfiguration": {  
    "types": [  
      "EDGE",  
      "REGIONAL"  
    ]  
  },  
  "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/  
cd833b28-58d2-407e-83e9-dce3fd852149",  
  "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"  
}
```

Für den migrierten regionalen benutzerdefinierten Domännennamen gibt die resultierende Eigenschaft `regionalDomainName` den regionalen API-Hostnamen zurück. Sie müssen einen DNS-Datensatz so einrichten, dass der regionale benutzerdefinierte Domänenname auf diesen regionalen Hostnamen verweist. Auf diese Weise kann der Datenverkehr, der an den benutzerdefinierten Domännennamen gebunden ist, an den regionalen Host weitergeleitet werden.

Nachdem Sie den DNS-Datensatz eingerichtet haben, können Sie den Edge-optimierten benutzerdefinierten Domainnamen entfernen. Der folgende [update-domain-name](#) Befehl entfernt den Edge-optimierten benutzerdefinierten Domännennamen:

```
aws apigateway update-domain-name \  
  --domain-name 'api.example.com' \  
  --patch-operations '[  
    { "op": "remove", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-  
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }  
  ]'
```

```
--domain-name api.example.com \  
--patch-operations '[  
    {"op":"remove", "path":"/endpointConfiguration/types", "value":"EDGE"},  
    {"op":"remove", "path":"/certificateName"},  
    {"op":"remove", "path":"/certificateArn"}  
'
```

Das folgende Verfahren zeigt, wie Sie einen Edge-optimierten benutzerdefinierten Domännennamen, der eine erweiterte Sicherheitsrichtlinie verwendet, zu einem regionalen benutzerdefinierten Domännennamen migrieren, der ebenfalls eine erweiterte Sicherheitsrichtlinie verwendet.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen Edge-optimierten benutzerdefinierten Domainnamen aus.
4. Klicken Sie unter Endpunktkonfiguration auf Bearbeiten.
5. Wählen Sie Regionalen Endpunkt hinzufügen aus.
6. Wählen Sie unter ACM-Zertifikat ein Zertifikat aus.

Das regionale Zertifikat muss sich in derselben Region befinden wie die regionale API.

7. Wählen Sie für Sicherheitsrichtlinie eine Sicherheitsrichtlinie aus, die mit `SecurityPolicy_` beginnt.
8. Wählen Sie für den Endpunktzugriffsmodus die Option Basic.
9. Wählen Sie Änderungen speichern aus.
10. Richten Sie einen DNS-Datensatz so ein, dass der regionale benutzerdefinierte Domainname auf diesen regionalen Hostnamen verweist. Weitere Informationen finden Sie unter [Konfiguration von Route 53 zur Weiterleitung von Datenverkehr an API Gateway](#).
11. Nachdem Sie bestätigt haben, dass Ihre DNS-Konfiguration den richtigen Endpunkt verwendet, löschen Sie die Edge-optimierte Endpunktkonfiguration. Wählen Sie Ihren benutzerdefinierten Domainnamen aus und klicken Sie dann unter Edge-optimierte Endpunktkonfiguration auf Löschen.
12. Bestätigen Sie Ihre Auswahl und löschen Sie den Endpunkt.

AWS CLI

Der folgende [update-domain-name](#) Befehl migriert einen Edge-optimierten benutzerdefinierten Domainnamen zu einem regionalen benutzerdefinierten Domainnamen:

```
aws apigateway update-domain-name \  
  --domain-name 'api.example.com' \  
  --patch-operations '[  
    { "op": "add", "path": "/endpointConfiguration/types", "value": "REGIONAL" },  
    { "op": "replace", "path": "/securityPolicy",  
      "value": "SecurityPolicy_TLS13_1_3_2025_09"},  
    { "op": "add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-  
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }  
  ]'
```

Das regionale Zertifikat muss sich in derselben Region befinden wie die regionale API.

Die Ausgabe sieht wie folgt aus:

```
{  
  "certificateArn": "arn:aws:acm:us-  
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",  
  "certificateName": "edge-cert",  
  "certificateUploadDate": "2017-10-16T23:22:57Z",  
  "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",  
  "domainName": "api.example.com",  
  "endpointConfiguration": {  
    "types": [  
      "EDGE",  
      "REGIONAL"  
    ]  
  },  
  "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",  
  "endpointAccessMode": "BASIC",  
  "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/  
cd833b28-58d2-407e-83e9-dce3fd852149",  
  "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"  
}
```

Für den migrierten regionalen benutzerdefinierten Domänennamen gibt die resultierende Eigenschaft `regionalDomainName` den regionalen API-Hostnamen zurück. Sie müssen einen DNS-Datensatz so einrichten, dass der regionale benutzerdefinierte Domänenname auf

diesen regionalen Hostnamen verweist. Auf diese Weise kann der Datenverkehr, der an den benutzerdefinierten Domännennamen gebunden ist, an den regionalen Host weitergeleitet werden.

Nachdem Sie den DNS-Datensatz eingerichtet haben, können Sie den Edge-optimierten benutzerdefinierten Domainnamen entfernen. Der folgende [update-domain-name](#) Befehl entfernt den Edge-optimierten benutzerdefinierten Domännennamen:

```
aws apigateway update-domain-name \
  --domain-name api.example.com \
  --patch-operations '[
    {"op":"remove", "path":"/endpointConfiguration/types", "value":"EDGE"},
    {"op":"remove", "path":"certificateName"},
    {"op":"remove", "path":"certificateArn"}
  ]'
```

Gehen Sie wie folgt vor, um einen regionalen benutzerdefinierten Domainnamen zu einem Edge-optimierten benutzerdefinierten Domainnamen zu migrieren.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie einen benutzerdefinierten Domainnamen aus.
4. Klicken Sie unter Endpunktkonfiguration auf Bearbeiten.
5. Klicken Sie auf Edge-optimierten Endpoint hinzufügen.
6. Wählen Sie unter ACM-Zertifikat ein Zertifikat aus.

Das Zertifikat der Edge-optimierten Domäne muss in der Region us-east-1 erstellt werden.

7. Wählen Sie Speichern.
8. Richten Sie einen DNS-Datensatz so ein, dass der Edge-optimierte benutzerdefinierte Domainname auf diesen Edge-optimierten Hostnamen verweist. Weitere Informationen finden Sie unter [Konfiguration von Route 53 zur Weiterleitung von Datenverkehr an API Gateway](#).
9. Nachdem Sie bestätigt haben, dass Ihre DNS-Konfiguration den richtigen Endpunkt verwendet, löschen Sie die regionale Endpunktkonfiguration. Wählen Sie Ihren benutzerdefinierten Domainnamen aus und klicken Sie dann unter Konfiguration des regionalen Endpunkts auf Löschen.

10. Bestätigen Sie Ihre Auswahl und löschen Sie den Endpunkt.

AWS CLI

Der folgende [update-domain-name](#) Befehl migriert Ihren regionalen benutzerdefinierten Domainnamen zu einem Edge-optimierten benutzerdefinierten Domainnamen:

```
aws apigateway update-domain-name \
  --domain-name 'api.example.com' \
  --patch-operations '[
    { "op": "add", "path": "/endpointConfiguration/types", "value": "EDGE" },
    { "op": "add", "path": "/certificateName", "value": "edge-cert" },
    { "op": "add", "path": "/certificateArn", "value": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a" }
  ]'
```

Das Zertifikat der Edge-optimierten Domäne muss in der Region us-east-1 erstellt werden.

Die Ausgabe sieht wie folgt aus:

```
{
  "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
  "certificateName": "edge-cert",
  "certificateUploadDate": "2017-10-16T23:22:57Z",
  "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
  "domainName": "api.example.com",
  "endpointConfiguration": {
    "types": [
      "EDGE",
      "REGIONAL"
    ]
  },
  "regionalCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/3d881b54-851a-478a-a887-f6502760461d",
  "regionalDomainName": "d-cgkq2qwgzf.execute-api.us-east-1.amazonaws.com"
}
```

Für den angegebenen benutzerdefinierten Domännennamen gibt API Gateway den Edge-optimierten API-Hostnamen als Wert der Eigenschaft `distributionDomainName` zurück. Sie müssen einen DNS-Datensatz so einrichten, dass der Edge-optimierte benutzerdefinierte

Domänenname auf diesen Verteilungsdomännennamen verweist. Auf diese Weise kann der Datenverkehr, der an den Edge-optimierten benutzerdefinierten Domännennamen gebunden ist, an den Edge-optimierten Hostnamen der API weitergeleitet werden.

Nachdem Sie den DNS-Datensatz eingerichtet haben, können Sie den Endpunkttyp REGION des benutzerdefinierten Domainnamens löschen. Mit dem folgenden [update-domain-name](#) Befehl wird der Endpunkttyp Regional entfernt:

```
aws apigateway update-domain-name \
  --domain-name api.example.com \
  --patch-operations '[
    {"op":"remove", "path":"/endpointConfiguration/types", value:"REGIONAL"},
    {"op":"remove", "path":"regionalCertificateArn"}
  ]'
```

Die Ausgabe sollte wie folgt aussehen:

```
{
  "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
  "certificateName": "edge-cert",
  "certificateUploadDate": "2017-10-16T23:22:57Z",
  "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
  "domainName": "api.example.com",
  "endpointConfiguration": {
    "types": "EDGE"
  }
}
```

Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie

Wenn Sie den Routing-Modus für Ihren benutzerdefinierten Domainnamen konfigurieren, legen Sie fest, wie eingehender Datenverkehr an Ihren weitergeleitet wird APIs. Sie senden Traffic APIs mithilfe von Routing-Regeln, API-Zuordnungen oder Routing-Regeln und API-Zuordnungen an Sie. Im folgenden Abschnitt wird erläutert, wann Sie Routing-Regeln verwenden sollten, wann API-Zuweisungen sinnvoll sind und wie Sie den Routing-Modus für Ihren benutzerdefinierten Domainnamen festlegen.

Wann ist die Verwendung von Routing-Regeln sinnvoll?

Wenn Sie Routing-Regeln verwenden, leiten Sie eingehende Anfragen, die bestimmte Bedingungen erfüllen, an bestimmte REST-Phasen weiter. APIs Beispielsweise kann eine Regel eine Anfrage an die `production`-Stufe Ihrer `users` REST-API weiterleiten, wenn sie den Header `version:v1` und den Basispfad `/users` enthält. Verwenden Sie Routing-Regeln, um erweiterte dynamische Routing-Topologien zu erstellen, die Anwendungsfälle wie das A/B Testen oder die zunehmende Nutzung neuer Versionen von Ihrem APIs unterstützen.

Wir empfehlen, beim Weiterleiten von Datenverkehr zu einer REST-API Routing-Regeln für den benutzerdefinierten Domainnamen zu verwenden. Sie können API-Zuweisungen mithilfe von Routing-Regeln neu erstellen. Weitere Informationen finden Sie unter [the section called “API-Zuweisung mithilfe von Routing-Regeln neu erstellen”](#).

Für REST APIs können Sie Routing-Regeln und API-Zuordnungen auch zusammen verwenden. Wenn Sie Routing-Regeln und API-Zuweisungen gemeinsam einsetzen, wertet API Gateway immer zuerst die Routing-Regeln aus, bevor es die API-Zuweisungen berücksichtigt. Verwenden Sie Routing-Regeln und API-Zuweisungen zusammen, um aktuelle benutzerdefinierte Domainnamen zu migrieren oder um Routing-Regeln auszuprobieren.

Überlegungen zu Routing-Regeln

Die folgenden Überlegungen können sich auf die Verwendung von Routing-Regeln auswirken:

- WebSocket oder HTTP APIs werden nicht als Ziel APIs für Routing-Regeln unterstützt.
- Wenn Ihr benutzerdefinierter Domainname API-Zuordnungen sowohl zu REST als auch zu HTTP enthält APIs, werden Routing-Regeln nicht unterstützt.
- Sie können eine Routing-Regel für eine private benutzerdefinierte Domain zu einer privaten REST-API erstellen. Ebenso können Sie eine Routing-Regel für eine öffentliche benutzerdefinierte Domain zu einer regionalen oder Edge-optimierten API erstellen.
- Sie können keine Routing-Regel für eine öffentliche benutzerdefinierte Domain zu einer privaten API erstellen. Ebenso können Sie keine Routing-Regel für eine private benutzerdefinierte Domain zu einer öffentlichen API erstellen.

Auswahl zwischen Routing-Regeln und API-Zuweisungen

Wir empfehlen, nach Möglichkeit Routing-Regeln zu verwenden. Verwenden Sie API-Zuordnungen nur, um Traffic an ein HTTP oder eine API zu senden. WebSocket

Festlegen des Routing-Modus für Ihren benutzerdefinierten Domainnamen

Sie können wählen, welchen Routing-Modus API Gateway verwendet, um den Datenverkehr an Ihren weiterzuleiten APIs. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#). In diesem Abschnitt werden Routing-Modi für benutzerdefinierte Domainnamen beschrieben. Sie müssen einen Routing-Modus für Ihren benutzerdefinierten Domainnamen festlegen, um den Datenverkehr an Ihren weiterzuleiten APIs. Folgende Routing-Modi werden unterstützt:

- `ROUTING_RULE_THEN_API_MAPPING` — Verwenden Sie diesen Modus, um Traffic APIs mit Routing-Regeln und API-Zuordnungen an Ihren zu senden. In diesem Modus haben alle Routing-Regeln Vorrang vor API-Zuweisungen. Ein Beispiel für diesen Modus finden Sie unter [the section called “Beispiel 2: Routing-Regeln und API-Zuweisungen”](#).
- `ROUTING_RULE_ONLY` — Verwenden Sie diesen Modus, um nur zu erlauben, dass Routing-Regeln Traffic an Ihren senden. APIs Wenn Ihr benutzerdefinierter Domainname diesen Modus verwendet, können Sie keine API-Zuordnung erstellen, aber Sie können den Befehl verwenden, um sie anzuzeigen. [get-api-mappings](#) API-Aufrufer können mit diesem Domainnamen keine API-Zuweisungen verwenden.
- `API_MAPPING_ONLY` — Verwenden Sie diesen Modus, um nur API-Zuordnungen zuzulassen, um Traffic an Ihren zu senden. APIs Wenn Ihr benutzerdefinierter Domainname diesen Modus verwendet, können Sie keine Routing-Regel erstellen, aber Sie können den Befehl `list-routing-rules` nutzen, um sie anzuzeigen. API-Aufrufer können mit diesem Domainnamen keine Routing-Regeln verwenden.

Dies ist der Standard-Routing-Modus für all vorhandenen Domainnamen und alle neuen Domainnamen, die Sie erstellen.

Wenn Sie einen benutzerdefinierten Domainnamen mit `apigateway` erstellen, wird `API_MAPPING_ONLY` als `BASE_PATH_MAPPING_ONLY` und `ROUTING_RULE_THEN_API_MAPPING` als `ROUTING_RULE_THEN_BASE_PATH_MAPPING` bezeichnet. Dieses Verhalten ist nur für, oder für alle vorhanden AWS CLI CloudFormation SDKs, nicht für. AWS-Managementkonsole

Das folgende Verfahren zeigt, wie Sie den Routing-Modus für einen vorhandenen benutzerdefinierten Domainnamen ändern. Wenn Sie den Routing-Modus ändern, können API-Aufrufer nicht über nicht unterstützte Routing-Modi auf Ihren Domainnamen zugreifen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen benutzerdefinierten Domainnamen aus.
4. Wählen Sie unter Domänendetails die Option Bearbeiten aus.
5. Wählen Sie für den Routing-Modus ROUTING_RULE_THEN_ API_MAPPING
6. Wählen Sie Speichern.

Wenn Sie den Routing-Modus in ROUTING_RULE_ONLY oder API_MAPPING_ONLY ändern, werden alle erstellten API-Zuweisungen oder Routing-Regeln von der Detailseite des Domainnamens in der Konsole entfernt. Wenn Sie den Routing-Modus so ändern, dass er entweder Routing-Regeln oder API-Zuweisungen unterstützt, werden diese Ressourcen wieder angezeigt.

AWS CLI - apigatewayv2

Mit dem folgenden [update-domain-name](#) Befehl wird ein Domainname aktualisiert, sodass er den Routing-Modus verwendet: ROUTING_RULE_THEN_API_MAPPING

```
aws apigatewayv2 update-domain-name \  
  --domain-name 'api.example.com' \  
  --routing-mode "ROUTING_RULE_THEN_API_MAPPING"
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "ApiMappingSelectionExpression": "$request.basepath",  
  "DomainName": "api.example.com",  
  "DomainNameArn": "arn:aws:apigateway:us-west-2::/domainnames/api.example.com",  
  "DomainNameConfigurations": [  
    {  
      "ApiGatewayDomainName": "d-abcdefg.execute-api.us-west-2.amazonaws.com",  
      "CertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/  
abcdefg-123456-abcdefg",  
      "DomainNameStatus": "AVAILABLE",  
      "EndpointType": "REGIONAL",
```

```

        "HostedZoneId": "Z20JLYMU09EFXC",
        "SecurityPolicy": "TLS_1_2"
    }
],
"RoutingMode": "ROUTING_RULE_THEN_API_MAPPING",
"Tags": {}
}

```

AWS CLI - apigateway

Mit dem folgenden [update-domain-name](#) Befehl wird ein privater benutzerdefinierter Domainname aktualisiert, sodass er den Routing-Modus verwendet `ROUTING_RULE_THEN_BASE_PATH_MAPPING`:

```

aws apigateway update-domain-name \
  --domain-name 'private.example.com' \
  --patch-operations "op='replace',path='/
routingMode',value='ROUTING_RULE_THEN_BASE_PATH_MAPPING'"

```

Die Ausgabe sieht wie folgt aus:

```

{
  "domainName": "private.example.com",
  "domainNameId": "abcd1234",
  "domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
  "certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
  "certificateUploadDate": "2024-09-10T10:31:20-07:00",
  "endpointConfiguration": {
    "types": [
      "PRIVATE"
    ],
    "ipAddressType": "dualstack"
  },
  "domainNameStatus": "AVAILABLE",
  "securityPolicy": "TLS_1_2",
  "policy": "...",
  "routingMode" : "ROUTING_RULE_THEN_BASE_PATH_MAPPING"
}

```

Routing-Regeln zum Verbinden von API-Stufen mit einem benutzerdefinierten Domainnamen für REST APIs

Eine Routing-Regel besteht aus einer Reihe von Bedingungen, bei deren Erfüllung eine Aktion ausgelöst wird. Eine Regel kann beispielsweise jede eingehende Anfrage, die an einen benutzerdefinierten Domainnamen gerichtet ist und den Header `Hello:World` sowie den Basispfad `users` enthält, an die `production`-Stufe einer REST-API weiterleiten.

Regeln werden in der Reihenfolge ihrer Priorität ausgewertet. Wenn Sie den Routing-Modus `ROUTING_RULE_THEN_API_MAPPING` verwenden, wertet API Gateway stets zuerst alle Routing-Regeln aus, bevor API-Zuweisungen geprüft werden. Die folgende Liste beschreibt, wie eine Routing-Regel Bedingungen, Aktionen und Prioritäten verwendet.

Bedingungen

Wenn die Bedingungen für eine Regel erfüllt sind, wird die dazugehörige Aktion durchgeführt. API Gateway unterstützt bis zu zwei Header-Bedingungen und eine Pfadbedingung. API Gateway wertet Header-Bedingungen und Basispfadbedingungen zusammen aus.

Sie können eine Regel auch ohne Bedingungen erstellen. Wenn API Gateway diese Regel auswertet, wird die Aktion immer ausgeführt. Eine Regel ohne Bedingungen kann als Catch-all-Regel verwendet werden.

Weitere Informationen über Header-Bedingungen finden Sie unter [the section called “Abgleichen der Header-Bedingungen”](#). Weitere Informationen über Pfadbedingungen finden Sie unter [the section called “Basispfadbedingungen anpassen”](#).

Aktionen

Aktionen sind das Ergebnis der Übereinstimmung von Bedingungen mit einer Routing-Regel. Derzeit wird nur das Aufrufen einer Stufe einer REST-API unterstützt.

Jede Regel kann eine Aktion haben.

Priorität

Die Priorität bestimmt die Reihenfolge, in der Regeln ausgewertet werden, ausgehend vom niedrigsten Wert hin zum höchsten Wert. Zwei Regeln können nicht dieselbe Priorität haben.

Sie können die Priorität im Bereich 1–1.000.000 festlegen. Eine Regel mit der Priorität 1 wird von API Gateway zuerst ausgewertet. Wir empfehlen, beim Erstellen einer Regel Lücken zwischen

den Prioritäten zu lassen. So können Sie die Reihenfolge später leichter ändern oder neue Regeln hinzufügen. Weitere Informationen finden Sie unter [the section called “Ändern der Priorität einer Routing-Regel”](#).

Beispiele dafür, wie API Gateway Routing-Regeln auswertet, finden Sie unter [the section called “Beispiele dafür, wie API Gateway Routing-Regeln auswertet”](#).

Bedingungstypen für Routing-Regeln in API Gateway

Im Folgenden werden die Bedingungstypen für Routing-Regeln beschrieben. API Gateway wendet eine Regel nur dann an, wenn alle Bedingungen erfüllt sind.

Ableichen der Header-Bedingungen

Wenn Sie eine Header-Bedingung erstellen, können Sie den Header-Namen und den Header-Glob-Wert abgleichen, z. B. `Hello:World`. API Gateway verwendet für den Abgleich von Header-Bedingungen eine exakte Übereinstimmung. Ihre Bedingung kann bis zu zwei Header mit AND zwischen ihnen verwenden. Eine Bedingung kann also z. B. dann übereinstimmen, wenn eine eingehende Anfrage sowohl `Hello:World` als auch `x-version:beta` enthält.

Bei der Übereinstimmung des Header-Namens wird die Groß-/Kleinschreibung nicht berücksichtigt, beim Header-Glob-Wert hingegen schon. `Hello:World` stimmt mit `hello:World` überein, jedoch nicht mit `Hello:world`.

Eine Liste der eingeschränkten Header-Werte finden Sie unter [the section called “Einschränkungen”](#).

Verwenden von Platzhaltern mit Header-Bedingungen

Platzhalter dürfen nur im Header-Glob-Wert verwendet werden. Zulässig sind dabei: `*prefix-match`, `*suffix-match*` oder `*contains*`. Die folgende Tabelle zeigt Beispiele, wie Platzhalter für die Übereinstimmung mit Header-Bedingungen verwendet werden können.

Header-Bedingungen	Anfragen, die der Routing-Regel entsprechen	Anfragen, die nicht der Routing-Regel entsprechen
<code>x-version: a*</code>	• <code>x-version: account</code> • <code>x-version: alpha</code>	• <code>x-version: backup</code> • <code>x-version: beta</code> • <code>x-version: users</code>

Header-Bedingungen	Anfragen, die der Routing-Regel entsprechen	Anfragen, die nicht der Routing-Regel entsprechen
<code>x-version: *a</code>	• <code>x-version: alpha</code> • <code>x-version: beta</code>	• <code>x-version: account</code> • <code>x-version: backup</code> • <code>x-version: users</code>
<code>x-version: *a*</code>	• <code>x-version: account</code> • <code>x-version: alpha</code> • <code>x-version: backup</code> • <code>x-version: beta</code>	• <code>x-version: users</code>
<code>x-version: *a*</code> und <code>x-version: *b*</code>	• <code>x-version: backup</code> • <code>x-version: beta</code>	• <code>x-version: account</code> • <code>x-version: alpha</code> • <code>x-version: users</code>
<code>x-version: b*</code> und <code>x-version: *a</code>	• <code>x-version: beta</code>	• <code>x-version: account</code> • <code>x-version: alpha</code> • <code>x-version: backup</code> • <code>x-version: users</code>
<code>x-version: *</code>	• <code>x-version: account</code> • <code>x-version: alpha</code> • <code>x-version: backup</code> • <code>x-version: beta</code> • <code>x-version: users</code>	Keine

Wenn Sie Bedingungen für mehrere Header-Werte erstellen, wie z. B. `Accept:application/json,text/xml`, empfehlen wir, `*contains*` für Ihre Header-Bedingungen zu verwenden und keine Bedingungen mit dem Kommazeichen (,) zu erstellen.

Da API Gateway Header-Bedingungen wörtlich prüft, können semantisch gleiche Anfragen unterschiedlich weitergeleitet werden. Die folgende Tabelle zeigt die unterschiedlichen Ergebnisse der Routing-Regeln.

Header-Bedingungen	Anfragen, die der Routing-Regel entsprechen	Anfragen, die nicht der Routing-Regel entsprechen
Accept: *json	- Accept:application/json - Accept:text/xml	Accept:application/json,text/xml
Accept: *json*	- Accept:application/json - Accept:text/xml - Accept:application/json,text/xml	Keine

Basispfadbedingungen anpassen

Wenn Sie eine Basispfadbedingung erstellen und die eingehende Anfrage den von Ihnen angegebenen Pfad enthält, wird die Regel angewendet. Bei der Übereinstimmung wird die Groß- und Kleinschreibung berücksichtigt, daher stimmt der Pfad `New/Users` nicht mit `new/users` überein.

Sie können eine Basispfadbedingung nur für einen Basispfad erstellen.

Eine Liste der eingeschränkten Basispfadbedingungen finden Sie unter [the section called "Einschränkungen"](#).

Entfernen des Basispfads mit Basispfadbedingungen

Wenn Sie eine Basispfadbedingung erstellen, können Sie wählen, ob der Basispfad entfernt werden soll. Wenn Sie den Basispfad entfernen, löscht API Gateway den eingehenden übereinstimmenden Basispfad, wenn es die Ziel-API aufruft. Dieses Verhalten entspricht dem einer API-Zuweisung. Wenn Sie den Basispfad nicht entfernen, leitet API Gateway den gesamten Basispfad an die Ziel-API weiter. Wir empfehlen, den Basispfad nur dann zu entfernen, wenn Sie eine API-Zuweisung neu erstellen.

Die folgende Tabelle zeigt Beispiele dafür, wie API Gateway die Bedingung zum Entfernen des Basispfads auswertet.

Bedingung	Leerzeichen des Basispfads entfernen	Eingehende Anfragen	Ergebnis
Wenn der Basispfad <code>PetStoreShopper/dogs</code> enthält	Wahr	GET <code>https://example.com/PetStoreShopper/dogs</code>	API Gateway ruft die GET-Methode der <code>/-</code> -Ressource auf.
Wenn der Basispfad <code>PetStoreShopper/dogs</code> enthält	Falsch	GET <code>https://example.com/PetStoreShopper/dogs</code>	API Gateway ruft die GET-Methode der <code>PetStoreShopper/dogs -</code> -Ressource auf.
Wenn der Basispfad <code>PetStoreShopper</code> enthält	Wahr	GET <code>https://example.com/PetStoreShopper/dogs</code>	API Gateway ruft die GET-Methode der <code>dogs</code> -Ressource auf.
Wenn der Basispfad <code>PetStoreShopper</code> enthält	Falsch	GET <code>https://example.com/PetStoreShopper/dogs</code>	API Gateway ruft die GET-Methode der <code>PetStoreShopper/dogs -</code> -Ressource auf.
Wenn der Basispfad <code>PetStoreShopper</code> enthält	Wahr	GET <code>https://example.com/PetStoreShopper?birds=available</code>	API Gateway ruft die GET-Methode der <code>/-</code> -Ressource mit dem Abfragezeichenfolgenparameter <code>birds=available</code> auf.
Wenn der Basispfad <code>PetStoreShopper</code> enthält	Falsch	GET <code>https://example.com/PetStoreShopper?</code>	API Gateway ruft die GET-Methode der <code>/PetStoreShopper -</code> -Ressource mit dem Abfrageze

Bedingung	Leerzeichen des Basispfads entfernen	Eingehende Anfragen	Ergebnis
		birds=available	ichenfolgenparameter birds=available auf.

Einschränkungen

- Die Ziel-API und der benutzerdefinierte Domainname müssen sich im selben AWS Konto befinden.
- Jede Regel kann eine Ziel-API haben.
- Sie können eine Routing-Regel nur für einen privaten benutzerdefinierten Domainnamen an eine private API und für einen öffentlichen benutzerdefinierten Domainnamen an eine öffentliche API erstellen. Sie können öffentliche und private Ressourcen nicht mischen.
- Wenn Ihr benutzerdefinierter Domainname API-Zuordnungen sowohl zu REST als auch zu HTTP enthält APIs, werden Routing-Regeln nicht unterstützt.
- Die maximal zulässige Priorität beträgt 1 000 000.
- Header-Einschränkungen:
 - Jede anyOf Bedingung darf nur einen Header-Wert enthalten.
 - Die einzigen zulässigen Zeichen für Header-Namen und Header-Glob-Werte sind in [RFC 7230](#) festgelegt. Dabei handelt es sich um a-z, A-Z, 0-9 und die folgenden Sonderzeichen: *? - ! # \$ % & ' . ^ _ ` | ~.
 - Sie können einen Platzhalter im Header-Glob-Wert verwenden, jedoch muss der Platzhalter *prefix-match, suffix-match* oder *contains* sein. Sie können * nicht in der Mitte eines Header-Glob-Werts verwenden.
 - Platzhalter-Header-Namen werden nicht unterstützt.
 - Der Header-Name darf maximal 40 Zeichen lang sein.
 - Der Header-Glob-Wert darf maximal 128 Zeichen lang sein.
 - Der Header-Glob-Wert für eine Infix-Übereinstimmung darf maximal 40 Zeichen lang sein.
 - Die folgenden Header werden nicht als Bedingungen unterstützt:
 - access-control-*
 - apigw-*
 - Authorization

- Connection
 - Content-Encoding
 - Content-Length
 - Content-Location
 - Forwarded
 - Keep-Alive
 - Origin
 - Proxy-Authenticate
 - Proxy-Authorization
 - TE
 - Trailers
 - Transfer-Encoding
 - Upgrade
 - x-amz-*
 - x-amzn-*
 - x-apigw-api-id
 - X-Forwarded-For
 - X-Forwarded-Host
 - X-Forwarded-Proto
 - x-restAPI
 - Via
 - Basispfadeinschränkungen:
 - Die Länge des Basispfads darf maximal 128 Zeichen betragen.
 - Der Basispfad darf nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: \$-_.+!*'()/.
- Diese Zeichen werden für reguläre Ausdrücke (Regex) nicht unterstützt.
- Der Basispfad darf nicht mit einem umgekehrten Schrägstrich (\) beginnen oder enden.

Beispiele dafür, wie API Gateway Routing-Regeln auswertet

Der folgende Abschnitt zeigt 4 Beispiele dafür, wie API Gateway Routing-Regeln und API-Zuweisungen auswertet.

Beispiel 1: Nur Routing-Regeln

In diesem Beispiel ist für den benutzerdefinierten Domainnamen `https://petstore.example.com` der Routing-Modus auf `ROUTING_RULE_ONLY` eingestellt und es gelten die folgenden Routing-Regeln und Prioritäten.

Regel-ID	Priorität	Bedingungen	Action
abc123	10	Wenn die Anfrage einen Header enthält: <code>Hello:World</code>	Ziel-API 1
zzz000	50	Wenn die Anfrage die folgenden Header enthält: <code>Accept:image/webp</code> und <code>Pet:Dog-*</code> und wenn der Basispfad <code>PetStoreShopper</code> enthält	Ziel-API 2
efg456	100	Keine	Ziel-API 3

Die folgende Tabelle zeigt, wie API Gateway die vorherigen Routing-Regeln auf Beispielanfragen anwendet.

Anfrage	Ausgewählte API	Erklärung
<code>https://petstore.example.com -h "Hello:World"</code>	Ziel-API 1	Die Anfrage entspricht der Routing-Regel abc123.

Anfrage	Ausgewählte API	Erklärung
<code>https://petstore.example.com/PetStoreShopper -h "Hello:World", "Pet:Dog-Bella", "Accept:image/webp"</code>	Ziel-API 1	API Gateway wertet alle Routing-Regeln in der Reihenfolge ihrer Priorität aus. Die Routing-Regel abc123 hat die höchste Priorität und die Bedingungen stimmen überein, daher ruft API Gateway die Ziel-API 1 auf. Obwohl die Bedingungen der Anfrage auch der Routing-Regel zzz000 übereinstimmen, wertet API Gateway nach einer Übereinstimmung keine weiteren Routing-Regeln aus.
<code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella", "Accept:image/webp"</code>	Ziel-API 2	Die Anfrage entspricht der Routing-Regel zzz000. Dies war eine Übereinstimmung, da Pet:Dog-Bella eine Zeichenfolgenübereinstimmung mit Pet:Dog-* war.
<code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella"</code>	Ziel-API 3	Die Anfrage entspricht nicht der Routing-Regel abc123. Die Anfrage entspricht nicht der Routing-Regel zzz000, da nicht alle erforderlichen Header vorhanden sind. Die nächste Prioritätsregel entspricht allen eingehenden Anfragen, daher ruft API Gateway die Ziel-API 3 auf.

Beispiel 2: Routing-Regeln und API-Zuweisungen

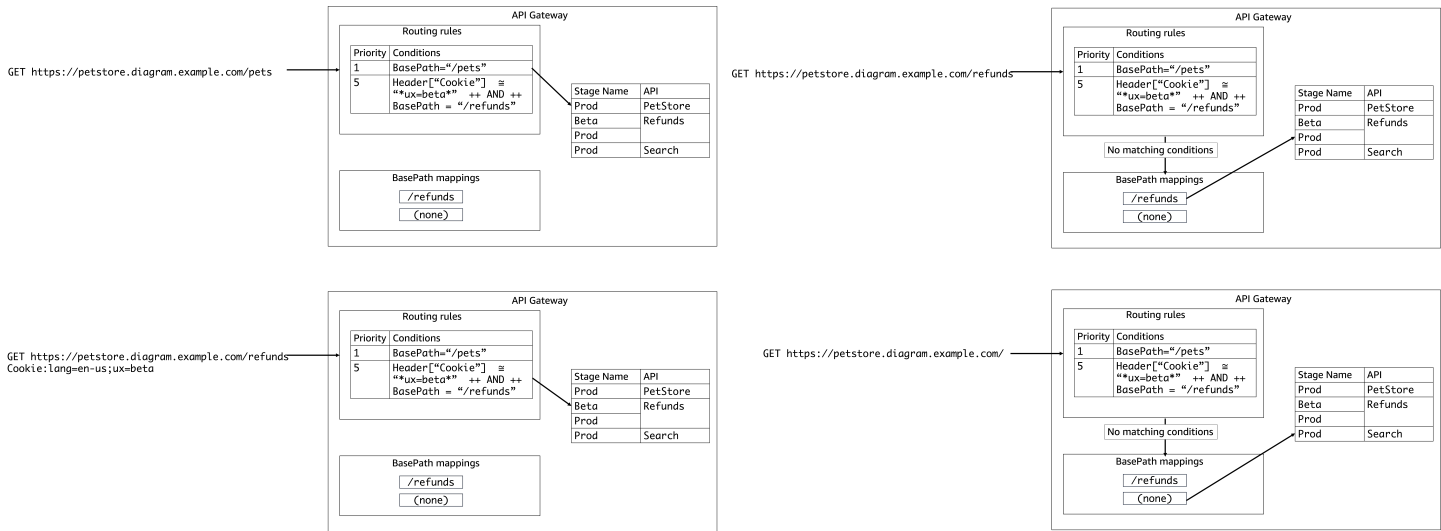
In diesem Beispiel ist für den benutzerdefinierten Domainnamen `https://petstore.diagram.example.com` der Routing-Modus auf `ROUTING_RULE_THEN_API_MAPPING` festgelegt und es gelten die folgenden Routing-Regeln und API-Zuweisungen.

Regel-ID	Priorität	Bedingungen	Action
abc123	1	Falls die Anfrage <code>pets</code> beinhaltet	Rufen Sie die Prod-Stufe der PetStore-API auf.
000zzz	5	Wenn die Anfrage die folgenden Header enthält: <code>Cookie:*ux=beta*</code> und wenn der Basispfad <code>/refunds</code> enthält	Rufen Sie die Beta-Stufe der Refunds-API auf.

Die folgende Tabelle zeigt die API-Zuweisungen für `https://petstore.backup.example.com`.

API-Zuweisungen	Ausgewählte API
<code>/refunds</code>	Rufen Sie die Prod-Stufe der Refunds-API auf.
(none)	Rufen Sie die Prod-Stufe der Search-API auf.

Das folgende Diagramm veranschaulicht, wie API Gateway die zuvor genannten Routing-Regeln und API-Zuweisungen auf Beispielanfragen anwendet. Die Beispielanfragen sind in der Tabelle nach diesem Diagramm zusammengefasst.



Die folgende Tabelle zeigt, wie API Gateway die vorherigen Routing-Regeln und API-Zuweisungen auf Beispielanfragen anwendet.

Anfrage	Ausgewählte API	Erklärung
<code>https://petstore.diagram.example.com/pets</code>	Die Prod-Stufe der PetStore-API.	Die Anfrage entspricht der Routing-Regel abc123.
<code>https://petstore.diagram.example.com/refunds -h "Cookie:lang=en-us;ux=beta"</code>	Die Beta-Stufe der Refunds-API.	Die Anfrage entspricht der Routing-Regel 000zzz. Der Cookie-Header enthält die korrekte <code>*contains*</code> - Übereinstimmung und die Basispfadübereinstimmung für diese Bedingung.
<code>https://petstore.diagram.example.com/refunds</code>	Die Prod-Stufe der Refunds-API.	Die Anfrage verfügt nicht über die erforderlichen Header, um mit der Routing-Regel zzz000 übereinzustimmen. Wenn API Gateway eine Routing-Regel nicht erfolgreich abgleichen kann, greift es auf API-Zuweisungen zurück. API Gateway kann den Basispfad

Anfrage	Ausgewählte API	Erklärung
		der Prod-Stufe der Refunds-API zuweisen.
<code>https://petstore.d iagram.example.com/</code>	Die Prod-Stufe der Search-API.	Die Anfrage stimmt mit der API-Zuweisung zum leeren Pfad (none) überein.

Beispiel 3: Routing-Regeln und API-Zuweisungen mit mehreren Ebenen

In diesem Beispiel ist für den benutzerdefinierten Domainnamen `https://petstore.backup.example.com` der Routing-Modus `ROUTING_RULE_THEN_API_MAPPING` festgelegt und es gelten die folgenden Routing-Regeln und API-Zuweisungen.

Die folgende Tabelle zeigt die Routing-Regeln für `https://petstore.backup.example.com`.

Regel-ID	Priorität	Bedingungen	Action
abc123	10	Wenn die Anfrage einen Header enthält: Hello:World	Ziel-API 1
000zzz	50	Wenn die Anfrage die folgenden Header enthält: Accept:image/webp und Pet:Dog-* und wenn der Basispfad PetStoreShopper enthält	Ziel-API 2

Die folgende Tabelle zeigt die API-Zuweisungen für `https://petstore.backup.example.com`.

API-Zuweisungen	Ausgewählte API
PetStoreShopper	Ziel-API 3
PetStoreShopper/cats	Ziel-API 4

Die folgende Tabelle zeigt, wie API Gateway die vorherigen Routing-Regeln und API-Zuweisungen auf Beispielanfragen anwendet.

Anfrage	Ausgewählte API	Erklärung
<code>https://petstore.example.com/PetStoreShopper -h "Accept:image/webp", "Pet:Cats"</code>	Ziel-API 3	Die Anfrage verfügt nicht über die erforderlichen Header, um mit der Routing-Regel zzz000 übereinzustimmen. Wenn API Gateway eine Routing-Regel nicht erfolgreich abgleichen kann, greift es auf API-Zuweisungen zurück. API Gateway kann den Basispfad der Ziel-API 3 zuweisen.
<code>https://petstore.example.com/PetStoreShopper/cats -h "Hello:World"</code>	Ziel-API 1	Die Anfrage entspricht der Routing-Regel abc123. Wenn der Routing-Modus auf <code>ROUTING_RULE_THEN_API_MAPPING</code> festgelegt ist, haben Routing-Regeln immer Vorrang vor API-Zuweisungen.
<code>https://petstore.example.com/Admin -h "Pet:Dog-Bella"</code>	Keine	Die Anfrage entspricht keiner Routing-Regel und keinen API-Zuweisungen. Da keine Standard-Routing-Regel vorliegt, lehnt API Gateway den Aufruf ab und sendet dem

Anfrage	Ausgewählte API	Erklärung
		Aufrufer den Statuscode 403 Forbidden .

Beispiel 4: Routing-Regeln für Platzhalter-Domainnamen

In diesem Beispiel ist der benutzerdefinierte Domainname `https://*.example.com` ein Platzhalter-Domainname. Der Platzhalter unterstützt alle Subdomains, die auf dieselbe Domain zurückgeleitet werden. Die folgenden Beispiel-Routing-Regeln ändern dieses Verhalten, sodass Subdomains mithilfe des Headers an ein anderes Ziel APIs weiterleiten können. Host

Die folgende Tabelle zeigt die Routing-Regeln für `https://*.example.com`.

Regel-ID	Priorität	Bedingungen	Action
abc123	10	Wenn die Anfrage einen Header enthält: Host : a . example . com	Ziel-API 1
000zzz	50	Wenn die Anfrage Header enthält: Host : b . example . com	Ziel-API 2
efg456	500	Keine	Ziel-API 3

Die folgende Tabelle zeigt, wie API Gateway die vorherigen Routing-Regeln auf Beispielanfragen anwendet.

Anfrage	Ausgewählte API	Erklärung
<code>https://a.example.com</code>	Ziel-API 1	Der Host-Header ist <code>a.example.com</code> . Diese Anfrage entspricht der Routing-Regel abc123.

Anfrage	Ausgewählte API	Erklärung
<code>https://b.example.com</code>	Ziel-API 2	Der Host-Header ist <code>b.example.com</code> . Diese Anfrage entspricht der Routing-Regel <code>000zzz</code> .
<code>https://testing.example.com</code>	Ziel-API 3	Dies entspricht der Auffang-Routing-Regel <code>efg456</code> .

Verwendung von Routing-Regeln

Sie können eine Routingregel mit dem AWS-Managementkonsole AWS CLI, oder einem beliebigen AWS SDK erstellen. Nachdem Sie eine Regel erstellt haben, können Sie deren Priorität ändern.

Erstellen einer Routing-Regel

Das folgende Verfahren zeigt, wie Sie eine Routing-Regel für einen benutzerdefinierten Domainnamen erstellen, dessen Routing-Modus entweder auf `ROUTING_RULE_THEN_API_MAPPING` oder `ROUTING_RULE_ONLY` festgelegt ist.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen benutzerdefinierten Domainnamen aus.
4. Wählen Sie auf der Registerkarte Routing-Details die Option Routing-Regel hinzufügen aus.
5. Wählen Sie Neue Bedingung hinzufügen aus, um eine neue Bedingung hinzuzufügen.

Sie können eine Header- oder Basispfadbedingung hinzufügen. Um alle eingehenden Anfragen Ihrem benutzerdefinierten Domainnamen zuzuweisen, fügen Sie keine Bedingung hinzu.

6. Verwenden Sie für Aktion das Dropdown-Menü, um Ihre Ziel-API und Ihre Zielstufe auszuwählen.
7. Wählen Sie Weiter aus.
8. Geben Sie im Prioritätsfeld eine Zahl für Ihre Priorität ein.

API Gateway wertet Regeln in der Reihenfolge der Priorität aus, ausgehend vom niedrigsten Wert bis hin zum höchsten Wert.

Wenn Sie eine Regel ohne Bedingung erstellen, empfehlen wir, eine Priorität mit hohem Wert zu verwenden.

9. Wählen Sie Routing-Regel erstellen aus.

AWS CLI

Mit dem folgenden [create-routing-rule](#) Befehl wird eine Routingregel mit der Priorität 50 erstellt. In diesem Beispiel leitet API Gateway alle eingehenden Anfragen, die die Header `Hello:World`, `x-version:beta` sowie den Basispfad `PetStoreShopper` enthalten, an die Ziel-API `a1b2c3` weiter.

```
aws apigatewayv2 create-routing-rule \  
  --domain-name 'api.example.com' \  
  --priority 50 \  
  --conditions '[  
    {  
      "MatchHeaders": {  
        "AnyOf": [  
          {  
            "Header": "Hello",  
            "ValueGlob": "World"  
          }  
        ]  
      }  
    },  
    {  
      "MatchHeaders": {  
        "AnyOf": [  
          {  
            "Header": "x-version",  
            "ValueGlob": "beta"  
          }  
        ]  
      }  
    },  
    {  
      "MatchBasePaths": {  
        "AnyOf": [  

```

```

        "PetStoreShopper"
    ]
}
]'\
--actions '[
{
    "InvokeApi": {
        "ApiId": "a1b2c3",
        "Stage": "prod"
    }
}
]'
```

Die Ausgabe sieht wie folgt aus.

```

{
  "Actions": [
    {
      "InvokeApi": {
        "ApiId": "a1b2c3",
        "Stage": "prod",
        "StripBasePath": false
      }
    }
  ],
  "Conditions": [
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "Hello",
            "ValueGlob": "World"
          }
        ]
      }
    },
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "x-version",
            "ValueGlob": "beta"
          }
        ]
      }
    }
  ]
}
```



```

    }
  ]
},
{
  "MatchBasePaths": {
    "AnyOf": [
      "PetStoreShopper"
    ]
  }
},
],
"Priority": 50,
"RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
"RoutingRuleId": "abc123"
}

```

Ändern der Priorität einer Routing-Regel

Sie können die Priorität einer Routing-Regel ändern. Dies tritt sofort in Kraft und kann beeinflussen, wie API-Benutzer Ihre benutzerdefinierten Domainnamen aufrufen. Wir empfehlen, beim Festlegen der Prioritäten Ihrer Routing-Regeln Lücken zwischen den Regeln zu lassen.

Stellen Sie sich beispielsweise zwei Routing-Regeln vor: Regel abc123 mit einer Priorität von 50 und Regel zzz000 mit einer Priorität von 150. Um die Priorität so zu ändern, dass API Gateway Regel zzz000 zuerst auswertet, können Sie die Priorität von Regel zzz000 auf 30 setzen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen benutzerdefinierten Domainnamen aus.
4. Wählen Sie auf der Registerkarte Routing-Details zunächst Ihre Routing-Regel und anschließend Bearbeiten aus.
5. Wählen Sie Weiter aus.
6. Geben Sie unter „Priorität“ die neue Priorität ein.
7. Wählen Sie Änderungen speichern aus.

AWS CLI

Der folgende [put-routing-rule](#) Befehl ändert die Priorität einer Routing-Regel. abc123

```
aws apigatewayv2 put-routing-rule \
  --domain-name 'api.example.com' \
  --priority 30 \
  --routing-rule-id abc123 \
  --conditions '[
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "Hello",
            "ValueGlob": "World"
          }
        ]
      }
    },
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "x-version",
            "ValueGlob": "beta"
          }
        ]
      }
    },
    {
      "MatchBasePaths": {
        "AnyOf": [
          "PetStoreShopper"
        ]
      }
    }
  ]' \
  --actions '[
    {
      "InvokeApi": {
        "ApiId": "a1b2c3",
        "Stage": "prod"
      }
    }
  ]'
```

```
]'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "Actions": [
    {
      "InvokeApi": {
        "ApiId": "a1b2c3",
        "Stage": "prod",
        "StripBasePath": false
      }
    }
  ],
  "Conditions": [
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "Hello",
            "ValueGlob": "World"
          }
        ]
      }
    },
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "x-version",
            "ValueGlob": "beta"
          }
        ]
      }
    },
    {
      "MatchBasePaths": {
        "AnyOf": [
          "PetStoreShopper"
        ]
      }
    }
  ]
},
```

```
"Priority": 38,
"RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
"RoutingRuleId": "abc123"
}
```

API-Zuweisung mithilfe von Routing-Regeln neu erstellen

Sie können eine API-Zuweisung mithilfe von Routing-Regeln neu erstellen. Um eine API-Zuweisung neu zu erstellen, stellen Sie sicher, dass das Entfernen des Basispfads aktiviert ist. Dadurch bleibt das Verhalten der API-Zuweisungen erhalten. Weitere Informationen finden Sie unter [the section called “Entfernen des Basispfads mit Basispfadbedingungen”](#).

Das folgende Tutorial zeigt, wie Sie die API-Zuweisung `https:// api.example.com/orders/v2/items/categories/5` als Routing-Regel neu erstellen. Außerdem erfahren Sie, wie Sie Ihre Zugriffsprotokolle aktualisieren, um die Routing-Regel-ID zu protokollieren, die API Gateway verwendet, um Datenverkehr an Ihre API zu senden.

AWS-Managementkonsole

Um den Routingmodus auf `ROUTING_RULE_THEN_` einzustellen `API_MAPPING`

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie Benutzerdefinierte Domainnamen im Hauptnavigationsbereich aus.
3. Wählen Sie Ihren benutzerdefinierten Domainnamen aus.
4. Wählen Sie unter Domänendetails die Option Bearbeiten aus.
5. Wählen Sie für den Routing-Modus `ROUTING_RULE_THEN_` `API_MAPPING`
6. Wählen Sie Speichern aus.

Nachdem Sie den Routing-Modus festgelegt haben, erstellen Sie die Routing-Regel.

So erstellen Sie die Routing-Regel

1. Wählen Sie auf der Registerkarte Routing-Details die Option Routing-Regel hinzufügen aus.
2. Wählen Sie Neue Bedingung hinzufügen und dann Pfad aus.
3. Geben Sie unter Pfad den Wert **`orders/v2/items/categories/5`** ein.

4. Wählen Sie für Leerzeichen des Basispfads entfernen die Option Aktiv aus.
5. Wählen Sie für Ziel-API Ihre Ziel-API aus.
6. Wählen Sie für Zielstufe Ihre Zielstufe aus.
7. Wählen Sie Weiter aus.
8. Geben Sie unter „Priorität“ eine Priorität ein.

Auch wenn Sie Ihre bestehende API-Zuweisung beibehalten, verwendet API Gateway immer die neue Routing-Regel, da Routing-Regeln stets Vorrang vor API-Zuweisungen haben.

9. Wählen Sie Änderungen speichern aus.

Nachdem Sie die Routing-Regel erstellt haben, aktualisieren Sie das Zugriffsprotokollformat für Ihre Stufe oder erstellen Sie ein neues Protokoll, um sicherzustellen, dass API Gateway Ihre Routing-Regel verwendet, um den Datenverkehr an Ihre API weiterzuleiten.

So aktualisieren Sie Ihre Zugriffsprotokolle

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API aus.
3. Klicken Sie im Hauptnavigationsbereich auf Stufen.
4. Wählen Sie unter Protokolle und Ablaufverfolgung die Option Bearbeiten aus.

Wenn Sie keine Protokollgruppe haben, siehe [the section called “CloudWatch protokolliert”](#).

5. Fügen Sie **\$context.customDomain.routingRuleIdMatched** Ihrem Protokollformat hinzu.

Diese Protokollgruppe zeichnet die Routing-Regel-ID auf, die API Gateway verwendet hat, um Datenverkehr an Ihre API zu senden. Weitere Informationen finden Sie unter [the section called “Ich kann nicht sagen, wie API Gateway Traffic an meinen gesendet hat APIs”](#).

6. Wählen Sie Speichern aus.

Nachdem Sie Ihre Zugriffsprotokolle aktualisiert haben, rufen Sie Ihren benutzerdefinierten Domainnamen auf. Das folgende Beispiel zeigt einen curl-Befehl zum Aufrufen des benutzerdefinierten Domainnamens `https://api.example.com` mit dem Basispfad `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

Nachdem Sie Ihren benutzerdefinierten Domainnamen erfolgreich aufgerufen haben, vergewissern Sie sich, dass CloudWatch Logs das anzeigt. `routingRuleIdMatched` Informationen zur Verwendung der CloudWatch Logs-Konsole zum Anzeigen einer Protokollgruppe finden Sie unter [the section called “Protokollereignis in der CloudWatch Konsole anzeigen”](#).

AWS CLI

1. Verwenden Sie den folgenden [update-domain-name](#) Befehl, um den Domainnamen `api.example.com` zu aktualisieren, um den Routing-Modus zu verwenden `ROUTING_RULE_THEN_API_MAPPING`.

```
aws apigatewayv2 update-domain-name \
  --domain-name 'api.example.com' \
  --routing-mode ROUTING_RULE_THEN_API_MAPPING
```

2. Verwenden Sie den folgenden [create-routing-rule](#) Befehl, um eine neue Routing-Regel zu erstellen, um die API-Zuordnung `https://api.example.com/orders/v2/items/categories/5` neu zu erstellen.

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'api.example.com' \
  --priority 50 \
  --conditions '[
  {
    "MatchBasePaths": {
      "AnyOf": [
        "orders/v2/items/categories/5"
      ]
    }
  }
]' \
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod",
      "StripBasePath": true
    }
  }
]
```

```
}  
]'
```

3. Verwenden Sie den folgenden Befehl [update-stage](#), um das Format der Zugriffsprotokolle so zu aktualisieren, dass die Variable `$context.customDomain.routingRuleIdMatched` enthalten ist. Diese Variable zeichnet die Routing-Regel-ID auf, die API Gateway verwendet hat, um Datenverkehr an Ihre API zu senden. Sie verwenden dieses Protokoll als Bestätigung, dass API Gateway Ihre Routing-Regel verwendet, um Datenverkehr an Ihre API zu senden. Weitere Informationen finden Sie unter [the section called “Ich kann nicht sagen, wie API Gateway Traffic an meinen gesendet hat APIs”](#).

```
aws apigateway update-stage \  
  --rest-api-id a1bc2c3 \  
  --stage-name prod \  
  --patch-operations "op=replace,path=/accessLogSettings/format,value='\  
$context.path \"$context.customDomain.routingRuleIdMatched \"$context.requestId \  
$context.extendedRequestId'"
```

Wenn Sie keine Protokollgruppe haben, siehe [the section called “CloudWatch protokolliert”](#).

4. Verwenden Sie den folgenden curl-Befehl, um Ihren benutzerdefinierten Domainnamen mit dem Basispfad `orders/v2/items/categories/5` aufzurufen.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

5. Verwenden Sie den folgenden [filter-log-events](#) Befehl, um die Protokollereignisse aus der Protokollgruppe abzurufen `access-log-group-orders`, die die Routingregel-ID `abc123` enthalten.

```
aws logs filter-log-events --log-group-name access-log-group-orders --filter-  
pattern abc123
```

Damit wird bestätigt, dass API Gateway die Routing-Regel verwendet hat, um den Datenverkehr an Ihre API zu senden.

Fehlerbehebung bei Problemen mit Routing-Regeln

Die folgende Anleitung zur Fehlerbehebung kann helfen, Probleme mit Ihren Routing-Regeln zu beheben.

Ich kann nicht sagen, wie API Gateway Traffic an meinen gesendet hat APIs

Sie können Zugriffsprotokolle für die Stufe Ihrer REST-API verwenden, um Ihre Routing-Regeln zu protokollieren und Fehler zu beheben. Mithilfe der Variablen `$context.customDomain.routingRuleIdMatched` können Sie die Routing-Regel-ID anzeigen, die API Gateway verwendet hat, um Datenverkehr an Ihre API zu senden. Um die API-Zuweisung anzuzeigen, die API Gateway verwendet hat, um Datenverkehr an Ihre API zu senden, verwenden Sie die Variable `$context.customDomain.basePathMatched`.

Um Ihre Routing-Regeln zu protokollieren, müssen Sie [einen entsprechenden CloudWatch Logs-Rollen-ARN](#) für Ihr Konto konfigurieren und eine Protokollgruppe erstellen.

Die folgende Beispielprotokollgruppe für Zugriffsprotokolle kann die relevanten Informationen zum Beheben von Problemen mit Routing-Regeln und API-Zuweisungen abrufen. API Gateway füllt die Kontextvariable nur für den Routing-Mechanismus aus, den es tatsächlich verwendet hat. Andernfalls ist die Kontextvariable -.

CLF

```
$context.path $context.customDomain.routingRuleIdMatched  
$context.customDomain.basePathMatched $context.requestId $context.extendedRequestId
```

JSON

```
{"requestPath": "$context.path", "routingRuleId" :  
  "$context.customDomain.routingRuleIdMatched", "API mapping" :  
  "$context.customDomain.basePathMatched", "requestId": "$context.requestId",  
  "extendedRequestId": "$context.extendedRequestId"}
```

XML

```
<request id="$context.requestId"> <requestPath>$context.path</requestPath>  
  <ruleId>$context.customDomain.routingRuleIdMatched</ruleId> <ApiMapping>  
  $context.customDomain.basePathMatched</ApiMapping> <extendedRequestId>  
  $context.extendedRequestId</extendedRequestId> </request>
```

CSV

```
$context.path,$context.customDomain.routingRuleIdMatched,  
$context.customDomain.basePathMatched,$context.requestId,$context.extendedRequestId
```


Wir empfehlen außerdem, den Routing-Modus für Ihren benutzerdefinierten Domainnamen zu bestätigen. Weitere Informationen finden Sie unter [the section called “Festlegen des Routing-Modus für Ihren benutzerdefinierten Domainnamen”](#).

Ich kann keine Routing-Regeln für meinen benutzerdefinierten Domainnamen aktivieren

Möglicherweise erhalten Sie den folgenden Fehler von API Gateway:

```
Your account doesn't have permission to use RoutingRules.  
This might be caused by an IAM policy in your account with a deny statement on  
BasePathMapping or ApiMapping.  
To grant permission for this account to use RoutingRules, use the UpdateAccount API.  
This will impact any existing IAM policies that deny access to BasePathMapping or  
ApiMapping.  
See API Gateway documentation for further details.
```

Sie erhalten diesen Fehler, wenn Sie eine IAM-Richtlinie haben oder hatten, die den Zugriff [BasePathMapping](#) auf oder verweigert. [ApiMapping](#) Wenn Sie Routing-Regeln für einen benutzerdefinierten Domainnamen aktivieren, wird der Zugriff auf BasePathMapping oder ApiMapping durch Ihre Richtlinie weiterhin verweigert. Dieselbe Richtlinie kann jedoch verwendet werden, um auf RoutingRule zuzugreifen. Dadurch könnte ein Benutzer das Routing-Verhalten Ihres benutzerdefinierten Domainnamens ändern.

Wenn Sie beispielsweise eine Richtlinie wie die folgende hatten:

```
{  
  "Sid": "DenyCreatingApiMappings",  
  "Effect": "Deny",  
  "Action": "apigateway:POST",  
  "Resource": [  
    "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"  
  ]  
}
```

Wenn Sie Routing-Regeln für `example.com` aktivieren, wird durch diese Richtlinie weiterhin der Zugriff auf das Erstellen einer ApiMapping verweigert, aber der Zugriff auf das Erstellen einer RoutingRule wird nicht verweigert.

Wir empfehlen, die IAM-Richtlinien in Ihrem Konto zu überprüfen. Die folgende Beispielrichtlinie verweigert den Zugriff auf das Erstellen von ApiMapping, BasePathMapping und RoutingRule:

```
{
  "Sid": "DenyCreatingBasePathMappingsApiMappings",
  "Effect": "Deny",
  "Action": "apigateway:POST",
  "Resource": [
    "arn:aws:apigateway:us-west-2::/domainnames/example.com/basepathmappings",
    "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
  ]
},
{
  "Sid": "DenyCreatingRoutingRules",
  "Effect": "Deny",
  "Action": "apigateway:CreateRoutingRule",
  "Resource": [
    "arn:aws:apigateway:us-west-2:111122223333:/domainnames/example.com/routingrules/*"
  ]
}
```

Nachdem Sie bestätigt haben, dass alle Ihre Richtlinien aktualisiert wurden, können Sie die kontoweiten Einstellungen Ihrer API aktualisieren, um Routing-Regeln für eine Region zu aktivieren.

Verwenden Sie dazu den folgenden [update-account](#)-Befehl, um die Einstellungen für die kontoweiten API-Einstellungen für eine Region zu aktualisieren:

```
aws apigateway update-account --patch-operations 'op=remove,path=/
features,value=BlockedForRoutingRules' --region us-west-2
```

Nachdem Sie die kontoweiten Einstellungen Ihrer API aktualisiert haben, können Sie den Routing-Modus Ihres benutzerdefinierten Domainnamens ändern. Sie können außerdem weiterhin IAM-Richtlinien verwenden, um den Zugriff auf `RoutingRules`, `ApiMapping` oder `BasePathMapping` zu verweigern.

Verwenden Sie API-Zuordnungen, um API-Stufen mit einem benutzerdefinierten Domainnamen für REST zu verbinden APIs

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Dadurch wird Traffic APIs über Ihren benutzerdefinierten Domainnamen an Sie gesendet.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise `https://api.example.com/orders` die production Stufe einer API zuordnen.

Sie können HTTP-API- und REST-API--Stufen demselben benutzerdefinierten Domain-Namen zuweisen.

Bevor Sie ein API-Mapping erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

Eingehende Anfragen an Ihren benutzerdefinierten Domainnamen

Wenn Sie einer Phase Ihrer API einen benutzerdefinierten Domainnamen zuordnen, entfernt API Gateway den eingehenden Basispfad. Dadurch wird der zugeordnete Basispfad aus dem API-Aufruf entfernt. Wenn Ihre Basispfad-Zuordnung beispielsweise der test Phase entspricht und Sie die folgende Anfrage verwenden `https://api.example.com/orders/shop/5/hats`, würde API Gateway die /hats Ressource der test Stufe Ihrer API aufrufen, nicht die orders/shop/5/hats Ressource. `https://api.example.com/orders/shop/5`

API-Anfragen zuordnen

Im Folgenden wird erklärt, wie API Gateway API-Zuordnungen auswertet.

Sie können eine API-Zuordnung mithilfe einstufiger Zuordnungen erstellen, z. B. einer API-Zuordnung von orders der beta Stufe einer API und einer API-Zuordnung von shipping der Stufe einer API. alpha Für regionale benutzerdefinierte Domainnamen mit der Sicherheitsrichtlinie TLS 1.2 unterstützt API Gateway mehrstufige API-Zuordnungen. Sie können eine API-Zuordnung zwischen orders/v1/items der alpha Stufe einer API und orders/v2/items der beta Phase einer API erstellen. Wenn Sie ein Mapping mit mehreren Ebenen erstellen, sendet API Gateway Anfragen an das API-Mapping, das den längsten passenden Pfad hat.

Sie können eine API-Zuordnung zu dem leeren Pfad erstellen (none). Wenn kein Pfad mit der Anfrage übereinstimmt, sendet API Gateway die Anfrage an den leeren Pfad (none).

In diesem Beispiel `https://api.example.com` hat der benutzerdefinierte Domainname die folgenden API-Zuordnungen.

API-Zuordnung	Ausgewählte API
(none)	API 1
orders	API 2
orders/v1/items	API 3
orders/v2/items	API 4
orders/v1/items/categories	API 5

Die folgende Tabelle zeigt, wie API Gateway die vorherigen API-Zuordnungen auf Beispielanfragen anwendet.

Anfrage	Ausgewählte API	Erklärung
<code>https://api.example.com/orders</code>	API 2	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v1/items</code>	API 3	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v2/items</code>	API 4	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v1/items/123</code>	API 3	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat. Das 123 am Ende der Anforderung hat keinen Einfluss auf die Auswahl. Siehe the section called “Eingehende Anfragen an

Anfrage	Ausgewählte API	Erklärung
		Ihren benutzerdefinierten Domainnamen ".
https://api.example.com/orders/v2/items/categories/5	API 5	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat.
https://api.example.com/customers	API 1	API Gateway verwendet das leere Mapping als Catch-All.
https://api.example.com/ordersandmore	API 2	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat. Bei einem benutzerdefinierten Domain-Namen, der mit einstufigen Mappings konfiguriert ist, z. B. nur https://api.example.com/orders und https://api.example.com/ , würde API Gateway API 1 auswählen, da es keinen passenden Pfad mit ordersandmore gibt.

Einschränkungen

- Bei einer API-Zuordnung APIs müssen sich der benutzerdefinierte Domainname und der zugeordnete Domainname im selben AWS Konto befinden.
- API-Mappings dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: \$ - _ . + ! * ' () / .
- Die maximale Länge für den Pfad in eines API-Mappings beträgt 300 Zeichen.

- Es können 200 API-Zuweisungen mit mehreren Ebenen für jeden Domainnamen vorhanden sein. Dieses Limit beinhaltet keine API-Zuordnung mit einzelnen Ebenen, wie /prod z.
- Mit der Sicherheitsrichtlinie TLS 1.2 können Sie HTTP APIs nur einem regionalen benutzerdefinierten Domainnamen zuordnen.
- Sie können nicht demselben benutzerdefinierten Domainnamen wie eine HTTP-API oder REST-API zuordnen. WebSocket APIs
- Sie müssen den Ressourceneintrag Ihres DNS-Anbieters erstellen oder aktualisieren, um ihn dem API-Endpunkt zuzuordnen, nachdem Sie Ihre API-Zuweisungen erstellt haben.
- Wenn Sie mehrstufige API-Zuweisungen erstellen, konvertiert API Gateway alle Header-Namen in Kleinbuchstaben.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Für Ihren benutzerdefinierten Domainnamen muss der Routing-Modus entweder auf `ROUTING_RULE_THEN_API_MAPPING` oder eingestellt sein `API_MAPPING_ONLY`. Informationen zum Einstellen des Routingmodus finden Sie unter [the section called “Festlegen des Routing-Modus für Ihren benutzerdefinierten Domainnamen”](#).

AWS Serverless Application Model Vorlagen, mit denen alle Ressourcen erstellt werden, finden Sie beispielsweise unter [Sitzungen mit aktiviertem SAM](#) GitHub.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Benutzerdefinierten Domänennamen im Hauptnavigationsbereich aus.
3. Wählen Sie einen benutzerdefinierten Domänennamen aus.
4. Wählen Sie auf der Registerkarte Routing-Details die Option API-Zuordnungen konfigurieren aus.
5. Geben Sie die API, die Stufe und den Pfad für die Zuweisung an.
6. Wählen Sie Speichern.

AWS CLI

Mit dem folgenden [create-api-mapping](#)-Befehl wird ein API-Mapping erstellt. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1/orders` an die angegebene API und Phase.

Note

Um API-Mappings mit mehreren Ebenen zu erstellen, müssen Sie `apigatewayv2` verwenden.

```
aws apigatewayv2 create-api-mapping \  
  --domain-name api.example.com \  
  --api-mapping-key v1/orders \  
  --api-id a1b2c3d4 \  
  --stage test
```

CloudFormation

Im folgenden CloudFormation Beispiel wird eine API-Zuordnung erstellt.

Note

Um API-Mappings mit mehreren Ebenen zu erstellen, müssen Sie `AWS::ApiGatewayV2` verwenden.

```
MyApiMapping:  
  Type: 'AWS::ApiGatewayV2::ApiMapping'  
  Properties:  
    DomainName: api.example.com  
    ApiMappingKey: 'orders/v2/items'  
    ApiId: !Ref MyApi  
    Stage: !Ref MyStage
```

IP-Adresstypen für benutzerdefinierte Domainnamen in API Gateway

Wenn Sie einen benutzerdefinierten Domainnamen erstellen, geben Sie den Typ der IP-Adressen an, über die Ihre Domain aufgerufen werden kann. Sie können wählen IPv4, ob IPv4 Adressen aufgelöst werden sollen, um Ihre Domain aufzurufen, oder Sie können Dualstack wählen, sodass IPv4 sowohl IPv6 Adressen als auch Adressen Ihre Domain aufrufen können. Wir empfehlen Ihnen, den IP-Adresstyp auf Dual-Stack einzustellen, um die Erschöpfung des IP-Speicherplatzes zu verringern oder um Ihre Sicherheitslage zu verbessern. [Weitere Informationen zu den Vorteilen eines Dual-Stack-IP-Adresstyps finden Sie unter. IPv6 AWS](#)

Sie können den IP-Adresstyp ändern, indem Sie die Endpunktkonfiguration Ihres Domainnamens aktualisieren.

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können sich auf Ihre Verwendung von IP-Adresstypen auswirken.

- Der Standard-IP-Adresstyp für benutzerdefinierte API Gateway Gateway-Domänennamen für die Öffentlichkeit APIs ist IPv4.
- Private benutzerdefinierte Domainnamen können nur einen Dual-Stack-IP-Adresstyp haben.
- Ihr benutzerdefinierter Domainname muss nicht für alle, die ihm APIs zugeordnet sind, denselben IP-Adresstyp haben. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies Auswirkungen darauf haben, wie Anrufer Ihre Domain aufrufen können.

Ändern Sie den IP-Adresstyp eines benutzerdefinierten Domainnamens

Sie können den IP-Adresstyp ändern, indem Sie die Endpunktkonfiguration des Domainnamens aktualisieren. Sie können die Endpunktkonfiguration aktualisieren, indem Sie das AWS-Managementkonsole, AWS CLI CloudFormation, oder ein AWS SDK verwenden.

AWS-Managementkonsole

Um den IP-Adresstyp eines benutzerdefinierten Domainnamens zu ändern

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie einen öffentlichen benutzerdefinierten Domainnamen.
3. Wählen Sie Endpunktkonfiguration.

4. Wählen Sie als IP-Adresstyp entweder IPv4 oder Dualstack aus.
5. Wählen Sie Speichern.

AWS CLI

Mit dem folgenden [update-domain-name](#) Befehl wird eine API aktualisiert, sodass sie den IP-Adresstyp Dualstack hat:

```
aws apigateway update-domain-name \  
  --domain-name dualstack.example.com \  
  --patch-operations "op='replace',path='/endpointConfiguration/  
ipAddressType',value='dualstack'"
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "domainName": "dualstack.example.com",  
  "certificateUploadDate": "2025-02-04T14:46:10-08:00",  
  "regionalDomainName": "d-abcd1234.execute-api.us-east-1.amazonaws.com",  
  "regionalHostedZoneId": "Z3LQWSYCGH4ADY",  
  "regionalCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/  
a1b2c3d4-5678-90ab-cdef",  
  "endpointConfiguration": {  
    "types": [  
      "REGIONAL"  
    ],  
    "ipAddressType": "dualstack"  
  },  
  "domainNameStatus": "AVAILABLE",  
  "securityPolicy": "TLS_1_2",  
  "tags": {}  
}
```

Wählen Sie eine Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus einer minimalen TLS-Version und Verschlüsselungssuites, die von API Gateway angeboten werden. Wenn Ihre Clients einen TLS-Handshake mit Ihrer API oder Ihrem benutzerdefinierten Domain-Namen erfordern, setzt die

Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite durch, die von API Gateway akzeptiert werden. Sicherheitsrichtlinien schützen Ihre APIs und benutzerdefinierte Domainnamen vor Netzwerksicherheitsproblemen wie Manipulation und Abhören zwischen einem Client und einem Server.

API Gateway unterstützt ältere Sicherheitsrichtlinien und erweiterte Sicherheitsrichtlinien.

TLS_1_0 und TLS_1_2 sind veraltete Sicherheitsrichtlinien. Verwenden Sie diese Sicherheitsrichtlinien für allgemeine Workloads oder um mit der Erstellung einer API zu beginnen. Jede Richtlinie, die mit `SecurityPolicy_` beginnt, ist eine erweiterte Sicherheitsrichtlinie. Sie können diese Richtlinien für regulierte Workloads, erweiterte Governance oder die Verwendung von Post-Quanten-Kryptographie verwenden. Wenn Sie eine optimierte Sicherheitsrichtlinie verwenden, müssen Sie auch den Endpunktzugriffsmodus festlegen, um zusätzliche Governance bereitzustellen. Weitere Informationen finden Sie unter [the section called “Endpunktzugriffsmodus”](#).

Überlegungen

Im Folgenden finden Sie Überlegungen zu Sicherheitsrichtlinien für benutzerdefinierte Domainnamen für REST APIs in API Gateway:

- Sie können Mutual TLS nicht für einen Domainnamen aktivieren, der eine erweiterte Sicherheitsrichtlinie verwendet.
- Sie können einem Domainnamen, der eine erweiterte Sicherheitsrichtlinie verwendet, keine HTTP-API zuordnen.
- Wenn Sie die mehrstufige Basispfadzuordnung zu einer REST-API aktivieren, die eine erweiterte Sicherheitsrichtlinie verwendet, können Sie keine Basispfadzuordnung zu einer HTTP-API für denselben Domainnamen erstellen.
- Ihre API kann einem benutzerdefinierten Domainnamen mit einer anderen Sicherheitsrichtlinie als Ihrer API zugeordnet werden. Wenn Sie diesen benutzerdefinierten Domainnamen aufrufen, verwendet API Gateway die Sicherheitsrichtlinie der API, um den TLS-Handshake auszuhandeln. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.
- API Gateway unterstützt Sicherheitsrichtlinien für alle APIs. Sie können jedoch nur eine Sicherheitsrichtlinie für REST auswählen APIs. API Gateway unterstützt nur die TLS_1_2 Sicherheitsrichtlinie für HTTP oder WebSocket APIs.
- API Gateway unterstützt nicht die Aktualisierung einer Sicherheitsrichtlinie für einen Domainnamen mit mehreren Endpunkttypen. Wenn Sie mehrere Endpunkttypen für einen Domainnamen haben, löschen Sie einen davon, um die Sicherheitsrichtlinie zu aktualisieren.

Wie API Gateway Sicherheitsrichtlinien anwendet

Das folgende Beispiel zeigt am Beispiel der Sicherheitsrichtlinie, wie API Gateway `SecurityPolicy_TLS13_1_3_2025_09` Sicherheitsrichtlinien anwendet.

Die `SecurityPolicy_TLS13_1_3_2025_09` Sicherheitsrichtlinie akzeptiert TLS 1.3-Verkehr und lehnt TLS 1.2- und TLS 1.0-Verkehr ab. Für TLS 1.3-Verkehr akzeptiert die Sicherheitsrichtlinie die folgenden Verschlüsselungssammlungen:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway akzeptiert keine anderen Cipher Suites. Zum Beispiel würde die Sicherheitsrichtlinie jeglichen TLS 1.3-Verkehr ablehnen, der die AES128-SHA Cipher Suite verwendet.

Um zu überwachen, welches TLS-Protokoll und welche Chiffren Clients für den Zugriff auf Ihr API Gateway verwendet haben, können Sie die `$context.cipherSuite` Kontextvariablen `$context.tlsVersion` und in Ihren Zugriffsprotokollen verwenden. Weitere Informationen finden Sie unter [the section called “Überwachen”](#).

Die Standardsicherheitsrichtlinien für alle REST APIs - und benutzerdefinierten Domainnamen finden Sie unter [the section called “Standard-Sicherheitsrichtlinien”](#). Informationen zu den unterstützten Sicherheitsrichtlinien für alle REST APIs - und benutzerdefinierten Domainnamen finden Sie unter [the section called “Unterstützte Sicherheitsrichtlinien”](#).

Ändern Sie die Sicherheitsrichtlinie Ihres benutzerdefinierten Domainnamens

Wenn Sie Ihre Sicherheitsrichtlinie ändern, dauert es etwa 15 Minuten, bis das Update abgeschlossen ist. Sie können den Wert `lastUpdateStatus` Ihres benutzerdefinierten Domainnamens überwachen. Wenn Ihr benutzerdefinierter Domainname aktualisiert wird, ist `lastUpdateStatus` er `PENDING` und wenn er abgeschlossen ist, wird er es auch sein `AVAILABLE`.

Wenn Sie eine Sicherheitsrichtlinie verwenden, die mit `beginntSecurityPolicy_`, müssen Sie auch den Endpunktzugriffsmodus aktivieren. Weitere Informationen finden Sie unter [the section called “Endpunktzugriffsmodus”](#).

AWS-Managementkonsole

Um die Sicherheitsrichtlinie eines benutzerdefinierten Domänennamens zu ändern

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie einen benutzerdefinierten Domainnamen, der Traffic an REST sendet. APIs
Stellen Sie sicher, dass Ihrem benutzerdefinierten Domainnamen nur ein Endpunkttyp zugeordnet ist.
3. Wählen Sie Benutzerdefinierte Domainnamen-Einstellungen und dann Bearbeiten aus.
4. Wählen Sie für Sicherheitsrichtlinie eine neue Richtlinie aus.
5. Wählen Sie für den Endpunktzugriffsmodus die Option Strict aus.
6. Wählen Sie Änderungen speichern aus.

AWS CLI

Mit dem folgenden [update-domain-name](#) Befehl wird ein Domainname aktualisiert, sodass er die SecurityPolicy_TLS13_1_3_2025_09 Sicherheitsrichtlinie verwendet:

```
aws apigateway update-domain-name \
  --domain-name example.com \
  --patch-operations '[
    {
      "op": "replace",
      "path": "/securityPolicy",
      "value": "SecurityPolicy_TLS13_1_3_2025_09"
    },
    {
      "op": "replace",
      "path": "/endpointAccessMode",
      "value": "STRICT"
    }
  ]'
```

Die Ausgabe sieht wie folgt aus:

```
{
  "domainName": "example.com",
  "endpointConfiguration": {
```

```
    "types": [ "REGIONAL" ],
    "ipAddressType": "dualstack"
  },
  "regionalCertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
  "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
  "endpointAccessMode": "STRICT"
}
```

Informationen zu HTTP APIs und WebSocket APIs

Weitere Hinweise zu HTTP APIs und WebSocket APIs finden Sie unter [the section called “Sicherheitsrichtlinie für HTTP APIs in API Gateway”](#) und [the section called “Sicherheitsrichtlinie für WebSocket APIs in API Gateway”](#).

Deaktivieren Sie den Standardendpunkt für REST APIs

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domännennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt. Clients können weiterhin eine Verbindung mit Ihrem Standardendpunkt herstellen, erhalten jedoch einen `403 Forbidden`-Statuscode. Die Deaktivierung des Standardendpunkts wirkt sich auf alle Stufen der API aus. Diese Einstellung wird wirksam, wenn Sie eine beliebige Einstellung auf einer Stufe aktualisieren, beispielsweise die Bereitstellung in der Stufe.

Im folgenden Vorgang sehen Sie, wie Sie den Standardendpunkt einer REST-API deaktivieren.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Hauptnavigationsbereich auf API-Einstellungen.
4. Wählen Sie eine API aus.
5. Klicken Sie unter API-Details auf Bearbeiten.
6. Wählen Sie unter Standardendpunkt die Option Inaktiv aus.
7. Wählen Sie Änderungen speichern aus.
8. Klicken Sie im Hauptnavigationsbereich auf Ressourcen.

9. Klicken Sie auf Deploy API.
10. Stellen Sie Ihre API erneut in einer Stufe bereit oder aktualisieren Sie eine beliebige Einstellung in einer Stufe, damit die Aktualisierung wirksam wird.

AWS CLI

Der folgende [update-rest-api](#)-Befehl deaktiviert den Standardendpunkt:

```
aws apigateway update-rest-api \  
  --rest-api-id abcdef123 \  
  --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird.

Der folgende [create-deployment](#)-Befehl erstellt eine Bereitstellung und ordnet sie einer Stufe zu:

```
aws apigateway create-deployment \  
  --rest-api-id abcdef123 \  
  --stage-name dev
```

Konfigurieren benutzerdefinierter Zustandsprüfungen für das DNS-Failover einer API-Gateway-API

Sie können Amazon Route 53-Zustandsprüfungen verwenden, um den DNS-Failover von einer API-Gateway-API in einer primären AWS-Region zu einer in einer sekundären Region zu kontrollieren. Dies kann dazu beitragen, die Auswirkungen im Falle eines regionalen Problems zu mildern. Wenn Sie eine benutzerdefinierte Domäne verwenden, können Sie ein Failover durchführen, ohne dass die Clients die API-Endpunkte ändern müssen.

Wenn Sie [Zielzustand bewerten](#) für einen Aliasdatensatz auswählen, schlagen diese Datensätze nur fehl, wenn der API-Gateway-Service in der Region nicht verfügbar ist. In einigen Fällen APIs kann es vor diesem Zeitpunkt zu einer Unterbrechung Ihres eigenen API Gateway kommen. Um das DNS-Failover direkt zu steuern, konfigurieren Sie benutzerdefinierte Route 53-Zustandsprüfungen für Ihr API Gateway APIs. In diesem Beispiel verwenden Sie einen CloudWatch Alarm, mit dem Betreiber das DNS-Failover steuern können. Weitere Beispiele und andere Überlegungen zur Konfiguration von Failover finden Sie unter [Erstellen von Notfallwiederherstellungsmechanismen mithilfe von Route](#)

[53](#) und [Durchführen von Route 53-Zustandsprüfungen für private Ressourcen in einer VPC mit AWS Lambda](#) und [CloudWatch](#)

Themen

- [Voraussetzungen](#)
- [Schritt 1: Einrichten der Ressourcen](#)
- [Schritt 2: Initiieren des Failovers zur sekundären Region](#)
- [Schritt 3: Testen des Failovers](#)
- [Schritt 4: Rückkehr zur Primärregion](#)
- [Nächste Schritte: Anpassen und regelmäßig testen](#)

Voraussetzungen

Um dieses Verfahren abzuschließen, müssen Sie die folgenden Ressourcen erstellen und konfigurieren:

- Eine Web-Domäne in Ihrem Besitz.
- Ein ACM-Zertifikat für diesen Domainnamen in zwei Teilen. AWS-Regionen Weitere Informationen finden Sie unter [the section called “Voraussetzungen für benutzerdefinierte Domainnamen”](#).
- Eine gehostete Route-53-Zone für Ihren Domänennamen. Weitere Informationen finden Sie unter [Arbeiten mit gehosteten Zonen](#) im Entwicklerhandbuch für Amazon Route 53.

Weitere Informationen zum Erstellen von Route-53-Failover-DNS-Einträgen für die Domain-Namen finden Sie unter [Eine Routing-Richtlinie auswählen](#) im Amazon-Route-53-Entwicklerhandbuch.

Weitere Informationen zur Überwachung eines CloudWatch Alarms finden Sie unter [Überwachung eines CloudWatch Alarms](#) im Amazon Route 53-Entwicklerhandbuch.

Schritt 1: Einrichten der Ressourcen

In diesem Beispiel erstellen Sie die folgenden Ressourcen, um das DNS-Failover für Ihren Domänennamen zu konfigurieren:

- API Gateway APIs in zwei Teilen AWS-Regionen
- API Gateway benutzerdefinierte Domainnamen mit demselben Namen in zwei AWS-Regionen
- API-Gateway-API-Zuordnungen, die Ihr API Gateway mit APIs den benutzerdefinierten Domainnamen verbinden

- Route-53-Failover-DNS-Einträge für die Domännennamen
- Ein CloudWatch Alarm in der sekundären Region
- Eine Route 53-Zustandsprüfung auf der Grundlage des CloudWatch Alarms in der sekundären Region

Stellen Sie zunächst sicher, dass Sie über alle erforderlichen Ressourcen in den primären und sekundären Regionen verfügen. Die sekundäre Region sollte den Alarm und die Zustandsprüfung enthalten. So sind Sie nicht von der primären Region abhängig, um das Failover durchzuführen. CloudFormation Vorlagen, die diese Ressourcen erstellen, finden Sie beispielsweise unter [primary.yaml](#) und [secondary.yaml](#).

 Important

Stellen Sie vor dem Failover in die sekundäre Region sicher, dass alle erforderlichen Ressourcen verfügbar sind. Andernfalls ist Ihre API nicht für den Datenverkehr in der sekundären Region bereit.

Schritt 2: Initiieren des Failovers zur sekundären Region

Im folgenden Beispiel empfängt die Standby-Region eine CloudWatch Metrik und leitet ein Failover ein. Wir verwenden eine benutzerdefinierte Metrik, bei der das Eingreifen des Bedieners erforderlich ist, um das Failover einzuleiten.

```
aws cloudwatch put-metric-data \  
  --metric-name Failover \  
  --namespace HealthCheck \  
  --unit Count \  
  --value 1 \  
  --region us-west-1
```

Ersetzen Sie die Metrikdaten durch die entsprechenden Daten für den von Ihnen CloudWatch konfigurierten Alarm.

Schritt 3: Testen des Failovers

Rufen Sie Ihre API auf und vergewissern Sie sich, dass Sie eine Antwort von der sekundären Region erhalten. Wenn Sie die Beispielvorgaben in Schritt 1 verwendet haben, ändert sich die Antwort von

„{"message": "Hello from the primary Region!"}“ nach dem Failover zu „{"message": "Hello from the secondary Region!"}“.

```
curl https://my-api.example.com
```

```
{"message": "Hello from the secondary Region!"}
```

Schritt 4: Rückkehr zur Primärregion

Um zur primären Region zurückzukehren, senden Sie eine CloudWatch Metrik, die dafür sorgt, dass die Integritätsprüfung bestanden wird.

```
aws cloudwatch put-metric-data \  
  --metric-name Failover \  
  --namespace HealthCheck \  
  --unit Count \  
  --value 0 \  
  --region us-west-1
```

Ersetzen Sie die Metrikdaten durch die entsprechenden Daten für den von Ihnen konfigurierten CloudWatch Alarm.

Rufen Sie Ihre API auf und vergewissern Sie sich, dass Sie eine Antwort von der primären Region erhalten. Wenn Sie die Beispielvorgaben in Schritt 1 verwendet haben, ändert sich die Antwort von „{"message": "Hello from the secondary Region!"}“ zu „{"message": "Hello from the primary Region!"}“.

```
curl https://my-api.example.com
```

```
{"message": "Hello from the primary Region!"}
```

Nächste Schritte: Anpassen und regelmäßig testen

Dieses Beispiel zeigt eine Möglichkeit, das DNS-Failover zu konfigurieren. Sie können eine Vielzahl von CloudWatch Metriken oder HTTP-Endpunkten für die Integritätsprüfungen zur Verwaltung des Failovers verwenden. Testen Sie Ihre Failover-Mechanismen regelmäßig, um sicherzustellen, dass sie erwartungsgemäß funktionieren und dass die Bediener mit Ihren Failover-Verfahren vertraut sind.

Optimieren Sie die Leistung von REST APIs

Nachdem Sie Ihre API für den Aufruf zur Verfügung gestellt haben, stellen Sie vielleicht fest, dass sie optimiert werden muss, um die Reaktionsfähigkeit zu verbessern. API Gateway bietet einige Strategien zur Optimierung Ihrer API, wie Antwort-Caching und Payload-Komprimierung. In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen aktivieren.

Themen

- [Cache-Einstellungen für REST APIs in API Gateway](#)
- [Payload-Komprimierung für REST APIs in API Gateway](#)

Cache-Einstellungen für REST APIs in API Gateway

Sie können API-Caching in Amazon API Gateway aktivieren, um die Antworten Ihres Endpunkts im Cache zu speichern. Mit Caching können Sie die Anzahl der an Ihren Endpunkt gemachten Aufrufe verringern und die Latenz der Anforderungen an Ihre API verbessern.

Wenn Sie das Caching für eine Phase aktivieren, speichert API Gateway Antworten von Ihrem Endpunkt für einen bestimmten Zeitraum time-to-live (TTL) in Sekunden zwischen. Anschließend beantwortet API Gateway die Anfrage, indem es die Endpunkt-Antwort aus dem Cache ausliest, anstatt eine Anfrage an Ihren Endpunkt zu senden. Der Standard-TTL-Wert für API-Caching ist 300 Sekunden. Die maximale TTL-Wert ist 3.600 Sekunden. TTL = 0 bedeutet, dass die Zwischenspeicherung deaktiviert ist.

Note

Caching ist die sinnvollste Tätigkeit. Sie können die CacheMissCount Metriken CacheHitCount und in Amazon verwenden CloudWatch , um Anfragen zu überwachen, die API Gateway aus dem API-Cache bedient.

Die maximale Größe einer Antwort, die zwischengespeichert werden kann, ist 1 048 576 Bytes. Die Cache-Datenverschlüsselung kann die Größe der Antwort erhöhen, wenn sie zwischengespeichert wird.

Dies ist ein HIPAA-berechtigter Service. [Weitere Informationen AWS zum US-amerikanischen Health Insurance Portability and Accountability Act von 1996 \(HIPAA\) und zur Nutzung von AWS Diensten](#)

[zur Verarbeitung, Speicherung und Übertragung geschützter Gesundheitsinformationen \(PHI\) finden Sie unter HIPAA Overview.](#)

 Important

Wenn Sie das Caching für eine Stufe aktivieren, wird standardmäßig nur die Zwischenspeicherung von GET-Methoden aktiviert. Dies hilft, die Sicherheit und die Verfügbarkeit Ihrer API zu gewährleisten. Sie können das Caching für andere Methoden [aktivieren, indem Sie die Methodeneinstellungen überschreiben](#).

 Important

Caching wird basierend auf der von Ihnen ausgewählten Cache-Größe nach Stunden berechnet. Caching kommt nicht für das kostenlose Kontingent in Frage. AWS Weitere Informationen finden Sie unter [API-Gateway-Preise](#).

Caching von Amazon API Gateway aktivieren

In API Gateway können Sie das Caching für eine bestimmte Stufe aktivieren.

Wenn Sie Caching aktivieren, müssen Sie eine Cache-Kapazität auswählen. Im Allgemeinen bietet eine größere Kapazität eine bessere Leistung, kostet aber auch mehr. Informationen zu den unterstützten Cachegrößen finden Sie [cacheClusterSize](#) in der API Gateway API Reference.

API Gateway ermöglicht das Caching durch die Erstellung einer dedizierten Cache-Instance. Dieser Vorgang kann bis zu 4 Minuten dauern.

API Gateway ändert die Caching-Kapazität durch Löschen der vorhandenen Cache-Instance und Erstellen einer neuen mit einer geänderten Kapazität. Alle Daten im Cache werden gelöscht.

 Note

Die Cache-Kapazität wirkt sich auf die CPU- und Netzwerkbandbreite der Cache-Instance aus. Infolgedessen kann sich die Cache-Kapazität auf die Performance Ihres Caches auswirken.

API Gateway empfiehlt, dass Sie einen 10-Minuten-Lastentlastungstest durchführen, um zu überprüfen, ob Ihre Cache-Kapazität für Ihre Arbeitslast geeignet ist. Stellen Sie sicher, dass

der Verkehr während des Lasttests den Produktionsverkehr widerspiegelt. Dazu gehören zum Beispiel Ramp Up, konstanter Verkehr und Verkehrsspitzen. Der Lasttest sollte Antworten enthalten, die aus dem Cache bereitgestellt werden können, sowie eindeutige Antworten, die dem Cache Elemente hinzufügen. Überwachen Sie die Latenz-, 4xx-, 5xx-, Cache-Hit- und Cache-Fehl-Metriken während des Belastungstests. Passen Sie Ihre Cache-Kapazität basierend auf diesen Metriken nach Bedarf an. Weitere Informationen zu Lasttests finden Sie unter [Wie wähle ich die beste Kapazität für den API-Gateway-Cache aus, um zu vermeiden, dass ein Ratenlimit erreicht wird?](#).

AWS-Managementkonsole

In der API-Gateway-Konsole konfigurieren Sie das Caching auf der Stages-Seite. Sie stellen den Stage-Cache bereit und legen eine Standard-Cache-Einstellung auf Methodenebene fest. Wenn Sie den Standard-Cache auf Methodenebene aktivieren, wird die Zwischenspeicherung auf Methodenebene für alle GET-Methoden der Stufe aktiviert, es sei denn, es wurde eine Methodenüberschreibung konfiguriert. Alle zusätzlichen GET-Methoden, die Sie in dieser Stufe bereitstellen, verfügen über einen Cache auf Methodenebene. Sie können Methodenüberschreibungen verwenden, um die Caching-Einstellungen auf Methodenebene für bestimmte Methoden Ihrer Stufe zu konfigurieren. Weitere Informationen zu Methodenüberschreibungen finden Sie unter [the section called “Überschreiben des Stufen-Caching für das Methoden-Caching”](#).

So konfigurieren Sie API-Caching für eine bestimmte Stufe:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Stages.
3. Wählen Sie in der API-Liste Stages den Namen der Stufe aus.
4. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
5. Aktivieren Sie unter Zusätzliche Einstellungen aktivieren Sie API-Cache bereitstellen für Zugriff auf Cache-Einstellungen.

Dadurch wird ein Cache-Cluster für Ihre Stufe bereitgestellt.

6. Aktivieren Sie Standard-Caching auf Methodenebene um für die Ebene Caching zuzulassen.

Dadurch wird das Caching auf Methodenebene für alle GET-Methoden dieser Stufe aktiviert. Alle zusätzlichen GET-Methoden, die Sie in dieser Stufe bereitstellen, werden dann über einen Cache auf Methodenebene verfügen.

 Note

Wenn bereits eine Einstellung für einen Cache auf Methodenebene vorhanden ist, wirkt sich eine Änderung des Standard-Cache auf Methodenebene nicht auf diese Einstellung aus.

Additional settings

Cache settings [Info](#)

You can enable API caching to cache your endpoint's responses. With caching, you can reduce the number of calls made to your endpoint and also improve the latency of requests to your API. Caching is charged by the hour based on cache size, see [API Gateway pricing](#) for details.

- ☒ **Provision API cache**
Provision API caching capabilities for your stage. Caching is not active until you enable the method-level cache.
- ☒ **Default method-level caching**
Activate method-level caching for all GET methods in this stage.

7. Wählen Sie **Änderungen speichern** aus.

AWS CLI

Der folgende [update-stage](#)-Befehl aktualisiert eine Stufe, um einen Cache bereitzustellen, und aktiviert das Methoden-Caching für alle GET-Methoden Ihrer Stufe.

```
aws apigateway update-stage \  
  --rest-api-id a1b2c3 \  
  --stage-name 'prod' \  
  --patch-operations file://patch.json
```

Der Inhalt von `patch.json` ist wie folgt:

```
[  
  {  
    "op": "replace",  
    "path": "/cacheClusterEnabled",
```

```
    "value": "true"
  },
  {
    "op": "replace",
    "path": "/cacheClusterSize",
    "value": "0.5"
  },
  {
    "op": "replace",
    "path": "/*/*/caching/enabled",
    "value": "true"
  }
]
```

Note

Wenn bereits eine Einstellung für einen Cache auf Methodenebene vorhanden ist, wirkt sich eine Änderung des Standard-Cache auf Methodenebene nicht auf diese Einstellung aus.

Note

Das API Gateway benötigt etwa 4 Minuten für das Erstellen und Löschen eines Caches. Bei Erstellung eines Cache springt der Cache-Cluster-Wert von `Create in progress` auf `Active`. Nach vollständiger Löschung des Cache springt der Cache-Cluster-Wert von `Delete in progress` auf `Inactive`.

Wenn Sie das Caching auf Methodenebene für alle Methoden in Ihrer Stufe aktivieren, springt der Wert für Standard-Cache auf Methodenebene auf `Active`. Wenn Sie das Caching auf Methodenebene für alle Methoden in Ihrer Stufe deaktivieren, springt der Wert für Standard-Cache auf Methodenebene auf `Inactive`. Wenn bereits eine Einstellung für Standard-Cache auf Methodenebene vorhanden ist, wirkt sich ein Statusänderung des Cache nicht auf diese Einstellung aus.

Wenn Sie das Caching in den Cache-Einstellungen einer Stufe aktivieren, werden nur GET-Methoden zwischengespeichert. Um die Sicherheit und Verfügbarkeit Ihrer API zu gewährleisten, empfehlen wir Ihnen, diese Einstellung nicht zu ändern. Sie können das Caching aber auch für andere Methoden [aktivieren, indem Sie die Methodeneinstellungen überschreiben](#).

Wenn Sie sich vergewissern möchten, ob die Zwischenspeicherung wie erwartet funktioniert, haben Sie zwei allgemeine Optionen:

- Untersuchen Sie die CloudWatch Metriken von CacheHitCount und CacheMissCount für Ihre API und Stage.
- Setzen Sie einen Zeitstempel in der Antwort.

 Note

Verwenden Sie nicht den X-Cache Header aus der CloudFront Antwort, um festzustellen, ob Ihre API von Ihrer API Gateway Gateway-Cache-Instance bedient wird.

Caching auf API-Gateway-Stufenebene für das Caching auf Methodenebene überschreiben

Sie können Cache-Einstellungen auf Stufenebene überschreiben, indem Sie das Caching für eine bestimmte Methode aktivieren oder deaktivieren. Sie können außerdem auch den TTL-Zeitraum ändern oder die Verschlüsselung für zwischengespeicherte Antworten ein- oder ausschalten.

Wenn Sie davon ausgehen, dass eine Methode sensible Daten in ihren Antworten erhält, verschlüsseln Sie Ihre Cache-Daten. Dies kann erforderlich sein, um verschiedene Compliance-Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

AWS-Managementkonsole

Wenn Sie die Einstellung für Standard-Caching auf Methodenebene in den Stufendetails ändern, hat dies keine Auswirkungen auf die Cache-Einstellungen auf Methodenebene, für die Überschreibungen konfiguriert sind.

Wenn Sie davon ausgehen, dass eine zwischengespeicherte Methode sensible Daten in ihren Antworten erhält, wählen Sie in den Cache-Einstellungen die Option Encrypt cache data aus.

So konfigurieren Sie API-Caching für einzelne Methoden mithilfe der Konsole:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie die API.

3. Wählen Sie Stages.
4. Erweitern Sie in der Liste Stufen für die API die Stufe, und wählen Sie eine Methode in der API aus.
5. Wählen Sie im Abschnitt Methodenüberschreibungen die Option Bearbeiten aus.
6. Aktivieren oder deaktivieren Sie im Abschnitt Methodeneinstellungen die Option Methoden-Cache aktivieren oder deaktivieren Sie die gewünschten Optionen.

**Note**

Das Caching ist erst dann aktiv, wenn Sie einen Cache-Cluster für Ihre Stufe bereitstellen.

7. Wählen Sie Speichern.

AWS CLI

Der folgende [update-stage](#)-Befehl deaktiviert den Cache nur für die GET /pets Methode:

```
aws apigateway update-stage /  
  --rest-api-id a1b2c3 /  
  --stage-name 'prod' /  
  --patch-operations file://patch.json
```

Der Inhalt von patch.json ist wie folgt:

```
[{  
  "op": "replace",  
  "path": "/~1pets/GET/caching/enabled",  
  "value": "false"  
}]
```

Verwenden der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren

Sie können eine Methode oder Integrationsparameter als Cache-Schlüssel verwenden, um zwischengespeicherte Antworten zu indizieren. Dazu gehören benutzerdefinierte Header, URL-Pfade oder Abfragezeichenfolgen. Sie können ein paar oder alle dieser Parameter als Cache-Schlüssel

festlegen - es muss jedoch mindestens ein Wert vorgegeben werden. Wenn ein Cache-Schlüssel vorhanden ist, speichert API Gateway die Antworten der einzelnen Schlüsselwerte separat im Cache; dasselbe geschieht wenn kein Cache-Schlüssel vorliegt.

 Note

Cache-Schlüssel sind beim Einrichten von Caching für eine Ressource erforderlich.

Angenommen, Sie haben eine Anforderung im folgenden Format:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

In dieser Anforderung kann `type` den Wert `admin` oder `regular` annehmen. Wenn Sie den `type`-Parameter als Teil des Cache-Schlüssels einbinden, werden die Antworten aus `GET /users?type=admin` getrennt von denen aus `GET /users?type=regular` zwischengespeichert.

Wenn eine Methode oder Integrationsanforderung mehr als einen Parameter annimmt, können Sie einige oder alle dieser Parameter einschließen, um den Cache-Schlüssel zu erstellen. Beispielsweise können Sie für die folgende Anforderung nur den `type`-Parameter in den Cache-Schlüssel einschließen, in der aufgelisteten Reihenfolge innerhalb eines TTL-Zeitraums:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

Die Antwort von dieser Anforderung wird im Cache gespeichert und verwendet, um die folgende Anforderung zu bedienen:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

AWS-Managementkonsole

Um einen Methoden- oder Integrationsanforderungsparameter als Teil eines Cache-Schlüssels in die API Gateway-Konsole aufzunehmen, wählen Sie Caching, nachdem Sie den Parameter hinzugefügt haben.

Edit method request

Method request settings
Authorization
None ▼
Request validator
None ▼
☐ API key required
Operation name - optional
GetPets

▼ URL query string parameters

Name	Required	Caching	
page	☐	☒	Remove
type	☐	☐	Remove

Add query string

AWS CLI

Der folgende [put-method](#)-Befehl erstellt eine GET-Methode und erfordert den Abfragezeichenfolgeparameter type:

```
aws apigateway put-method /  
  --rest-api-id a1b2c3 /  
  --resource-id aaa111 /  
  --http-method GET /  
  --authorization-type "NONE" /  
  --request-parameters "method.request.querystring.type=true"
```

Der folgende [put-integration](#)-Befehl erstellt eine Integration für die GET-Methode mit einem HTTP-Endpunkt und gibt an, dass API Gateway den Methodenanforderungsparameter `type` zwischenspeichert:

```
aws apigateway put-integration /
  --rest-api-id a1b2c3 /
  --resource-id aaa111 /
  --http-method GET /
  --type HTTP /
  --integration-http-method GET /
  --uri 'https://example.com' /
  --cache-key-parameters "method.request.querystring.type"
```

Um einen Integrationsanforderungsparameter für den API-Gateway-Cache festzulegen, verwenden Sie `integration.request.location.name` als Cache-Schlüsselparameter.

API-Stufen-Cache in API Gateway leeren

Wenn API-Zwischenspeicherung aktiviert ist, können Sie den Cache Ihrer API-Stufe löschen, um sicherzustellen, dass die Clients Ihrer API die neuesten Antworten von Ihren Integrationsendpunkten erhalten.

AWS-Managementkonsole

So leeren Sie den API-Stufen-Cache

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine API aus, die über eine Stufe mit einem Cache verfügt.
3. Wählen Sie zunächst im Hauptnavigationsbereich Stufen und anschließend Ihre Stufe mit einem Cache aus.
4. Wählen Sie das Menü Stufenaktionen und dann Stage-Cache bereinigen aus.

AWS CLI

Mit dem folgenden [flush-stage-cache](#)-Befehl wird der Stage-Cache geleert:

```
aws apigateway flush-stage-cache \
  --rest-api-id a1b2c3 \
```

```
--stage-name prod
```

Note

Nachdem der Cache gelöscht wurde, werden Antworten aus dem Integrationsendpunkt bedient, bis der Cache wieder aufgebaut ist. Während dieses Zeitraums kann sich die Anzahl der Anforderungen an den Integrationsendpunkt erhöhen. Dadurch kann sich die Gesamt-Latenz Ihrer API zeitweilig erhöhen.

API Gateway-Cache-Eintrag ungültig machen

Ein Client Ihrer API kann einen vorhandenen Cache-Eintrag entwerfen und ihn vom Integrationsendpunkt für einzelne Anforderungen neu laden. Der Client muss eine Anforderung senden, die den Header `Cache-Control: max-age=0` enthält. Der Client erhält die Antwort direkt vom Integrationsendpunkt anstelle des Zwischenspeichers, wenn der Client hierfür eine entsprechende Berechtigung hat. Dies ersetzt den vorhandenen Cache-Eintrag durch die neue Antwort, die vom Integrationsendpunkt abgerufen wird.

Um einem Aufrufer eine Berechtigung zu erteilen, fügen Sie an die IAM-Ausführungsrolle für den Client eine Richtlinie im folgenden Format an.

Note

Die kontoübergreifende Cache-Invalidierung wird nicht unterstützt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
```

```

    "arn:aws:execute-api:us-east-1:111111111111:api-id/stage-name/
    GET/resource-path-specifier"
  ]
}

```

Diese Richtlinie erlaubt es dem API Gateway-Ausführungsdienst, den Cache für Anfragen zur angegebenen Ressource (oder den angegebenen Ressourcen) ungültig zu machen. Um eine Gruppe zum Ziel gesetzter Ressourcen anzugeben, verwenden Sie ein Platzhalterzeichen (*) für *account-id*, *api-id* und andere Angaben im ARN-Wert von *Resource*. Weitere Informationen über das Festlegen von Berechtigungen für den API Gateway-Ausführungsdienst finden Sie unter [Zugriffskontrolle für eine API mit IAM-Berechtigungen](#).

Wenn sie keine `InvalidateCache`-Richtlinie angeben (oder das Kontrollkästchen `Autorisierung erforderlich` in der Konsole aktivieren), kann jeder beliebige Client den API-Zwischenspeicher entwerten. Wenn die meisten oder alle Clients den API-Cache entwerten, kann dies zu einer erheblichen Steigerung der Latenz Ihrer API führen.

Wenn die Richtlinie eingerichtet ist, wird das Caching aktiviert und eine Autorisierung ist erforderlich.

Sie können festlegen, wie API Gateway mit nicht autorisierten Anfragen umgehen soll, indem Sie aus den folgenden Optionen wählen:

Anfrage mit HTTP-Statuscode 403 ablehnen

API Gateway gibt eine `403 Unauthorized`-Antwort zurück.

Geben Sie zum Einstellen dieser Option über die API `FAIL_WITH_403`.

Cache-Control-Header ignorieren; Warnung im Antwort-Header hinzufügen

API Gateway verarbeitet die Anfrage und fügt im Antwort-Header eine Warnung hinzu.

Geben Sie zum Einstellen dieser Option über die API `SUCCESS_WITH_RESPONSE_HEADER`.

Cache-Control-Header ignorieren

API Gateway verarbeitet die Anfrage, fügt jedoch keine Warnung im Antwort-Header hinzu.

Geben Sie zum Einstellen dieser Option über die API `SUCCESS_WITHOUT_RESPONSE_HEADER`.

Sie können das Verhalten für nicht autorisierte Anfragen über die API-Gateway-Konsole oder die AWS CLI einstellen.

AWS-Managementkonsole

So geben Sie an, wie nicht autorisierte Anfragen behandelt werden sollen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API aus, die über eine Stufe mit einem Cache verfügt.
3. Wählen Sie zunächst im Hauptnavigationsbereich Stufen und anschließend Ihre Stufe mit einem Cache aus.
4. Wählen Sie unter Stufendetails die Option Bearbeiten aus.
5. Wählen Sie unter Bearbeitung nicht autorisierter Anfragen eine Option aus.
6. Klicken Sie auf Weiter.
7. Überprüfen Sie Ihre Änderungen und klicken Sie dann auf Änderungen speichern.

AWS CLI

Der folgende [update-stage](#)-Befehl aktualisiert eine Stufe, sodass nicht autorisierte Anfragen mit einem HTTP-Statuscode 403 abgelehnt werden:

```
aws apigateway update-stage /  
  --rest-api-id a1b2c3 /  
  --stage-name 'prod' /  
  --patch-operations 'op=replace,path=/*/*/caching/  
  unauthorizedCacheControlHeaderStrategy,value="FAIL_WITH_403"'
```

CloudFormation Beispiel für eine Phase mit einem Cache

Die folgende CloudFormation Vorlage erstellt eine Beispiel-API, stellt einen 0.5 GB-Cache für die Prod Stufe bereit und aktiviert das Caching auf Methodenebene für alle Methoden. GET

⚠ Important

Caching wird basierend auf der von Ihnen ausgewählten Cache-Größe nach Stunden berechnet. Caching ist für das kostenlose AWS -Kontingent nicht zulässig. Weitere Informationen finden Sie unter [API-Gateway-Preise](#).

CloudFormation Beispielvorlage

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: cache-example
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
  ApiStage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
```

```
StageName: Prod
Description: Prod Stage with a cache
RestApiId: !Ref Api
DeploymentId: !Ref ApiDeployment
CacheClusterEnabled: True
CacheClusterSize: 0.5
MethodSettings:
  - ResourcePath: /*
    HttpMethod: '*'
    CachingEnabled: True
```

Payload-Komprimierung für REST APIs in API Gateway

API Gateway ermöglicht es Ihrem Client, über eine der [unterstützten Inhaltskodierungen](#) Ihre API mit komprimierten Payloads aufzurufen. API Gateway unterstützt standardmäßig die Dekomprimierung der Methodenanforderungsnutzlast. Sie müssen Ihre API allerdings so konfigurieren, dass die Komprimierung der Methodenantwortnutzlast aktiviert wird.

Zum Aktivieren der Komprimierung für eine [API](#) legen Sie für die Eigenschaft [minimumCompressionsSize](#) beim oder nach dem Erstellen der API eine nicht negative Ganzzahl zwischen 0 und 10485760 (10 M Bytes) fest. Zum Deaktivieren der Komprimierung in der API legen Sie `minimumCompressionSize` auf Null fest oder entfernen Sie die Eigenschaft. Sie können die Komprimierung für eine API aktivieren oder deaktivieren, indem Sie die API Gateway Gateway-Konsole AWS CLI, die oder die API Gateway Gateway-REST-API verwenden.

Wenn Sie die Komprimierung auf eine Nutzlast jeder beliebigen Größe angewendet werden soll, legen Sie den Wert `minimumCompressionSize` auf Null fest. Durch die Komprimierung einer kleinen Datengröße kann sich jedoch die endgültige Datengröße erhöhen. Darüber hinaus kann die Komprimierung in API Gateway und die Dekomprimierung im Client die Gesamtlatenzzeit erhöhen und mehr Rechenzeit erfordern. Führen Sie Testläufe für Ihre API aus, um den optimalen Wert zu bestimmen.

Der Client kann eine API-Anfrage mit einer komprimierten Nutzlast und einem geeigneten Content-Encoding-Header für API Gateway senden, um geeignete Zuweisungsvorlagen zu komprimieren und anzuwenden, bevor die Anfrage dem Integrationsendpunkt übergeben wird. Nachdem die Komprimierung aktiviert und die API bereitgestellt wurde, kann der Client eine API-Antwort mit einer komprimierten Nutzlast empfangen, wenn ein geeigneter Accept-Encoding-Header in der Methodenanforderung angegeben ist.

Wenn der Integrationsendpunkt nicht komprimierte JSON-Nutzlasten erwartet und zurückgibt, ist jede Zuweisungsvorlage, die für eine nicht komprimierte JSON-Nutzlast konfiguriert ist, für die komprimierte Nutzlast anwendbar. Für eine komprimierte Methodenanforderungsnutzlast dekomprimiert API Gateway die Nutzlast, wendet die Zuweisungsvorlage an und übergibt dem Integrationsendpunkt die zugewiesene Anfrage. Für eine nicht komprimierte Methodenanforderungsnutzlast wendet API Gateway die Zuweisungsvorlage an, komprimiert die zugewiesene Nutzlast und gibt die komprimierte Nutzlast an den Client zurück.

Themen

- [Nutzlastkomprimierung für eine API in API Gateway aktivieren](#)
- [Aufrufen einer API-Methode mit einer komprimierten Nutzlast in API Gateway](#)
- [Empfangen einer API-Antwort mit einer komprimierten Nutzlast in API Gateway](#)

Nutzlastkomprimierung für eine API in API Gateway aktivieren

Sie können die Komprimierung für eine API mithilfe der API Gateway Gateway-Konsole AWS CLI, des oder eines AWS SDK aktivieren.

Im Fall einer vorhandenen API müssen Sie nach der Aktivierung der Komprimierung die API bereitstellen, damit die Änderungen wirksam werden. Im Fall einer neuen API können Sie die API nach Abschluss der API-Einrichtung bereitstellen.

Note

Die Inhaltskodierung mit der höchsten Priorität muss von API Gateway unterstützt werden. Wenn dies nicht der Fall ist, wird keine Komprimierung auf die Antwortnutzlast angewendet.

Themen

- [Payload-Komprimierung für eine API über die API Gateway-Konsole aktivieren](#)
- [Aktivieren Sie die Payload-Komprimierung für eine API mithilfe von AWS CLI](#)
- [Von API Gateway unterstützte Inhaltskodierungen](#)

Payload-Komprimierung für eine API über die API Gateway-Konsole aktivieren

Im folgenden Verfahren wird beschrieben, wie Sie die Nutzlastkomprimierung für eine API aktivieren.

So aktivieren Sie die Payload-Komprimierung über die API Gateway-Konsole:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.
3. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Aktivieren Sie die Inhaltskodierung, um die Nutzlast-Komprimierung zu aktivieren. Geben Sie einen Wert für die Mindestkomprimierungsgröße (in Byte) unter Mindesttextgröße ein. Um die Komprimierung zu deaktivieren, schalten Sie die Option Inhaltskodierung aus.
6. Wählen Sie Änderungen speichern aus.

Aktivieren Sie die Payload-Komprimierung für eine API mithilfe von AWS CLI

Der folgende [create-rest-api](#)Befehl erstellt eine API mit Payload-Komprimierung:

```
aws apigateway create-rest-api \  
  --name "My test API" \  
  --minimum-compression-size 0
```

Der folgende [update-rest-api](#)Befehl aktiviert die Payload-Komprimierung für eine bestehende API:

```
aws apigateway update-rest-api \  
  --rest-api-id 1234567890 \  
  --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

Die Eigenschaft `minimumCompressionSize` weist eine nicht negative ganze Zahl zwischen 0 und 10485760 (10 M Bytes) auf. Mit ihr wird die Komprimierung gemessen. Wenn die Nutzlastgröße kleiner als dieser Wert ist, wird die Komprimierung oder Dekomprimierung nicht auf die Nutzlast angewendet. Wenn er auf Null festgelegt ist, ist die Komprimierung für jede beliebige Nutzlastgröße erlaubt.

Der folgende [update-rest-api](#)Befehl deaktiviert die Payload-Komprimierung:

```
aws apigateway update-rest-api \  
  --rest-api-id 1234567890 \  
  --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

```
--patch-operations op=replace,path=/minimumCompressionSize,value=
```

Sie können `value` auch auf eine leere Zeichenfolge `""` festlegen oder die Eigenschaft `value` im vorhergehenden Aufruf vollständig weglassen.

Von API Gateway unterstützte Inhaltskodierungen

API Gateway unterstützt die folgenden Inhaltskodierungen:

- `deflate`
- `gzip`
- `identity`

API Gateway unterstützt auch das folgende `Accept-Encoding` Header-Format gemäß der Spezifikation [RFC 7231](#):

- `Accept-Encoding: deflate, gzip`
- `Accept-Encoding:`
- `Accept-Encoding: *`
- `Accept-Encoding: deflate; q=0.5, gzip; q=1.0`
- `Accept-Encoding: gzip; q=1.0, identity; q=0.5, *; q=0`

Aufrufen einer API-Methode mit einer komprimierten Nutzlast in API Gateway

Um eine API-Anfrage mit einer komprimierten Nutzlast zu erstellen, muss der Client den `Content-Encoding`-Header mit einer der [unterstützten Inhaltskodierungen](#) festlegen.

Angenommen, Sie sind ein API-Client und möchten die PetStore API-Methode (`POST /pets`) aufrufen. Rufen Sie die Methode nicht mit der folgenden JSON-Ausgabe auf:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

{
  "type": "dog",
  "price": 249.99
}
```

Stattdessen können Sie die Methode mit der gleichen Nutzlast aufrufen, die mit der GZIP-Codierung komprimiert wird:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

◆◆◆RPP*◆,HU◆RPJ◆0W◆◆e&◆◆◆L,◆,-y◆j
```

Wenn API Gateway die Anfrage erhält, prüft es, ob die angegebene Inhaltskodierung unterstützt wird. Anschließend wird versucht, die Nutzlast mit der angegebenen Inhaltskodierung zu dekomprimieren. Wenn die Dekomprimierung erfolgreich ist, wird die Anforderung an den Integrationsendpunkt gesendet. Wenn die angegebene Kodierung nicht unterstützt wird oder der gelieferte Payload nicht mit der angegebenen Kodierung komprimiert ist, gibt API Gateway die 415 `Unsupported Media Type`-Fehlerantwort zurück. Der Fehler wird nicht in CloudWatch Logs protokolliert, wenn er in der frühen Phase der Dekomprimierung auftritt, bevor Ihre API und Ihre Phase identifiziert wurden.

Empfangen einer API-Antwort mit einer komprimierten Nutzlast in API Gateway

Beim Erstellen einer Anforderung in einer komprimierungsfähigen API kann der Client festlegen, dass er eine komprimierte Antwortnutzlast eines bestimmten Formats erhält, indem er einen `Accept-Encoding`-Header mit einer [unterstützten Inhaltskodierung](#) angibt.

API Gateway komprimiert den Antwortpayload nur, wenn die folgenden Bedingungen erfüllt sind:

- Der `Accept-Encoding`-Header der eingehenden Anforderung weist eine unterstützte Inhaltskodierung und ein entsprechendes Format auf.

Note

Wenn der Header nicht festgelegt ist, ist der Standardwert `*` gemäß [RFC 7231](#). In einem solchen Fall komprimiert API Gateway den Payload nicht. Einige Browser oder Clients fügen komprimierungsfähigen Anforderungen möglicherweise automatisch `Accept-Encoding` (z. B. `Accept-Encoding:gzip, deflate, br`) hinzu. Dies kann die Komprimierung der Nutzlast in API Gateway aktivieren. Ohne explizite Angabe der unterstützten `Accept-Encoding`-Header-Werte komprimiert API Gateway den Payload nicht.

- Die `minimumCompressionSize` wird in der API festgelegt, um die Komprimierung zu aktivieren.

- Die Integrationsantwort enthält keinen Content-Encoding-Header.
- Die Größe einer Integrationsantwortnutzlast nach Anwendung der geltenden Zuweisungsvorlage ist größer als oder gleich dem angegebenen minimumCompressionSize-Wert.

API Gateway wendet ggf. eine Zuweisungsvorlagen an, die für die Integrationsantwort konfiguriert ist, bevor die Nutzlast komprimiert wird. Wenn die Integrationsantwort einen Content-Encoding-Header enthält, geht API Gateway davon aus, dass die Integrationsantwortnutzlast bereits komprimiert ist und überspringt die Komprimierungsverarbeitung.

Ein Beispiel ist das PetStore API-Beispiel und die folgende Anfrage:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

Das Backend beantwortet die Anforderung mit einer nicht komprimierten JSON-Nutzlast ähnlich wie der folgenden:

```
200 OK

[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Zum Empfangen dieser Ausgabe als komprimierte Nutzlast kann Ihr API-Client eine Anforderung wie folgt senden:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Der Client erhält die Antwort mit einem Content-Encoding-Header und einer GZIP-kodierten Nutzlast ähnlich wie den folgenden:

```
200 OK
Content-Encoding:gzip
...

RP

J)V
:P^IeA*+++++(L XXYZku0L0B7!9C#&++++Yaa^^X
```

Wird die Antwortnutzlast komprimiert, wird nur die komprimierte Datengröße für die Datenübertragung in Rechnung gestellt.

Verteilen Sie Ihren REST APIs an Kunden in API Gateway

In diesem Abschnitt finden Sie Einzelheiten zur Verteilung Ihres API Gateway APIs an Ihre Kunden. Die Verteilung Ihrer API umfasst die Generierung SDKs für Ihre Kunden zum Herunterladen und Integrieren in ihre Client-Anwendungen, die Dokumentation Ihrer API, damit Kunden wissen, wie sie sie von ihren Client-Anwendungen aus aufrufen können, und die Bereitstellung Ihrer API als Teil von Produktangeboten.

Themen

- [Nutzungspläne und API-Schlüssel für REST APIs in API Gateway](#)
- [Dokumentation für REST APIs in API Gateway](#)
- [Generieren Sie SDKs für REST APIs in API Gateway](#)
- [Verkaufen Sie Ihr API Gateway APIs über AWS Marketplace](#)

Nutzungspläne und API-Schlüssel für REST APIs in API Gateway

Nachdem Sie Ihre erstellt, getestet und bereitgestellt haben APIs, können Sie API Gateway Gateway-Nutzungspläne verwenden, um sie Ihren Kunden als Produktangebote zur Verfügung zu stellen. Sie

können Nutzungspläne und API-Schlüssel so konfigurieren, dass Kunden auf ausgewählte Pakete zugreifen können APIs, und damit beginnen, Anfragen auf der APIs Grundlage definierter Limits und Kontingente auf diese einzuschränken. Diese können auf API- oder API-Methodenebene festgelegt werden.

Was sind Nutzungspläne und API-Schlüssel?

Ein Nutzungsplan gibt an, wer Zugriff auf eine oder mehrere bereitgestellte API-Stufen und Methoden hat. Es wird optional die Zielerforderrate festgelegt, um die Drosselungsrate der Anfragen zu starten. Der Plan verwendet API-Schlüssel zur Identifizierung von API-Clients und derjenigen, welche auf die zugehörigen API-Stufen jedes Schlüssels zugreifen können.

API-Schlüssel sind alphanumerische Zeichenfolgenwerte, mit denen Sie Anwendungsentwicklerkunden Zugriff auf Ihre API gewähren. Sie können API-Schlüssel zusammen mit [Lambda-Autorisierern](#), [IAM-Rollen](#) oder [Amazon Cognito verwenden, um den Zugriff auf Ihre zu kontrollieren](#). APIs API Gateway kann API-Schlüssel für Sie erstellen. Alternativ können Sie Schlüssel aus einer [CSV-Datei](#) importieren. Sie können einen API-Schlüssel in API Gateway erstellen oder ihn aus einer externen Quelle in API Gateway importieren. Weitere Informationen finden Sie unter [the section called “API-Schlüssel für REST APIs in API Gateway einrichten”](#).

Ein API-Schlüssel verfügt über einen Namen und einen Wert. (Die Begriffe „API-Schlüssel“ und „API-Schlüsselwert“ werden häufig synonym verwendet.) Namen dürfen nicht mehr als 1024 Zeichen enthalten. Der Wert ist eine alphanumerische Zeichenfolge mit 20 bis 128 Zeichen, beispielsweise, `apikey1234abcdefghij0123456789`.

Important

API-Schlüsselwerte müssen eindeutig sein. Wenn Sie versuchen, zwei API-Schlüssel mit unterschiedlichen Namen und demselben Wert zu erstellen, hält API Gateway sie für ein- und denselben API-Schlüssel.

Ein API-Schlüssel kann mehreren Nutzungsplänen zugeordnet werden. Ein Nutzungsplan kann mehreren Stufen zugeordnet werden. Ein bestimmter API-Schlüssel kann jedoch nur einem Nutzungsplan für jede Stufe Ihres API zugeordnet werden.

Ein -Drosselungslimits legt den Zielpunkt fest, an dem die Anforderungsdrosselung beginnen soll. Dies kann auf API- oder API-Methodenebene festgelegt werden.

Ein Kontingentlimit legt die maximale Zielanzahl der Anforderungen fest, die mit einem bestimmten API-Schlüssel innerhalb eines vorgegebenen Zeitraums übermittelt werden können. Sie können einzelnen API-Methoden so konfigurieren, dass der API-Schlüssel autorisiert werden muss (abhängig von der Konfiguration des Nutzungsplans).

Die Drosselungs- und Kontingentlimits gelten für Anforderungen einzelner API-Schlüssel, die über alle API-Stufen eines Nutzungsplans aggregiert werden.

Note

Die -Drosselungsrate und die Kontingente der Nutzungspläne sind keine harten Grenzen und werden Best-Effort-Basis angewendet. In einigen Fällen können Kunden die von Ihnen festgelegten Kontingente überschreiten. Verlassen Sie sich nicht auf Nutzungsplankontingente oder Drosseln, um Kosten zu kontrollieren oder den Zugriff auf eine API zu blockieren. Überlegen Sie sich, [AWS Budgets](#) zu verwenden, um Kosten zu überwachen und [AWS WAF](#), um API-Anfragen zu verwalten.

Bewährte Methoden für API-Schlüssel und Nutzungspläne

Im Folgenden finden Sie empfohlene bewährte Methoden für die Verwendung von API-Schlüsseln und Nutzungsplänen.

Important

- Verwenden Sie keine API-Schlüssel zur Authentifizierung oder Autorisierung, um den Zugriff auf Ihre zu kontrollieren. APIs Wenn Sie mehrere APIs in einem Nutzungsplan haben, kann ein Benutzer mit einem gültigen API-Schlüssel für eine API in diesem Nutzungsplan auf alle APIs APIs in diesem Nutzungsplan zugreifen. Verwenden Sie für die Zugriffskontrolle auf Ihre API stattdessen eine IAM-Rolle, [einen Lambda-Genehmiger](#) oder einen [Amazon Cognito-Benutzerpool](#).
- Verwenden Sie API-Schlüssel, die API Gateway generiert. API-Schlüssel sollten keine vertraulichen Informationen enthalten. Clients übertragen sie normalerweise in Headern, die protokolliert werden können.

- Wenn Sie ein Entwicklerportal verwenden, um Ihre zu veröffentlichen APIs, beachten Sie, dass alle Ihre Daten APIs in einem bestimmten Nutzungsplan von Kunden abonniert werden können, auch wenn Sie sie für Ihre Kunden nicht sichtbar gemacht haben.
- In einigen Fällen können Kunden die von Ihnen festgelegten Kontingente überschreiten. Verlassen Sie sich nicht auf Nutzungspläne, um die Kosten zu kontrollieren. Ziehen Sie es in Betracht, [AWS Budgets](#) zu verwenden, um Kosten zu überwachen und [AWS WAF](#), um API-Anfragen zu verwalten.
- Der Aktualisierungsvorgang nach dem Hinzufügen eines API-Schlüssels zu einem Nutzungsplan kann ein paar Minuten dauern.

Auswählen einer API-Schlüsselquelle in API Gateway

Wenn Sie einen Nutzungsplan einer API zuordnen und API-Schlüssel bei API-Methoden aktivieren, muss jede eingehende API-Anfrage einen [API-Schlüssel](#) enthalten. API Gateway liest den Schlüssel und vergleicht ihn mit den Schlüsseln im Nutzungsplan. Wenn die Schlüssel übereinstimmen, werden die Anforderungen entsprechend der Anforderungslimits und des Kontingents des Plans von API Gateway gedrosselt. Andernfalls wird eine `InvalidKeyParameter`-Ausnahme ausgelöst. Dadurch erhält der Aufrufer eine 403 Forbidden-Antwort.

Ihre API Gateway-API kann API-Schlüssel aus einer von zwei Quellen erhalten:

HEADER

Sie verteilen API-Schlüssel an Ihre Kunden und erfordern die Übermittlung des API-Schlüssels als X-API-Key-Header bei jeder eingehenden Anforderung.

AUTHORIZER

Sie lassen einen Lambda-Genehmiger den API-Schlüssel im Rahmen der Autorisierungsantwort zurückgeben. Weitere Informationen zur Autorisierungsantwort finden Sie unter [the section called “Ausgabe von einem API-Gateway-Lambda-Genehmiger”](#).

Note

Zu berücksichtigende bewährte Methoden finden Sie unter [the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”](#).

Das folgende Verfahren zeigt, wie Sie eine API-Schlüsselquelle für eine API auswählen.

AWS-Managementkonsole

So wählen Sie eine API-Schlüsselquelle für eine API aus:

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.
3. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Wählen Sie unter API-Schlüsselquelle die Option Header oder Authorizer aus der Drop-down-Liste aus.
6. Wählen Sie Änderungen speichern aus.

AWS CLI

Mit dem folgenden [update-rest-api](#) Befehl wird eine API aktualisiert, sodass die API-Schlüsselquelle wie folgt festgelegt wird: `AUTHORIZER`:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations  
op=replace,path=/apiKeySource,value=AUTHORIZER
```

Damit der Client einen API-Schlüssel übermittelt, setzen Sie im vorherigen Befehl `value` auf `HEADER`.

REST API

Wenn Sie eine API-Schlüsselquelle für eine API über die API Gateway-REST-API auswählen möchten, rufen Sie wie folgt [restapi:update](#) auf:

```
PATCH /restapis/fugvjdxtri/ HTTP/1.1  
Content-Type: application/json  
Host: apigateway.us-east-1.amazonaws.com  
X-Amz-Date: 20160603T205348Z  
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160603/us-east-1/  
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,  
Signature={sig4_hash}  
  
{  
  "patchOperations" : [  
    {
```

```
    "op" : "replace",
    "path" : "/apiKeySource",
    "value" : "HEADER"
  }
]
```

Damit ein Genehmiger einen API-Schlüssel zurückgibt, legen Sie den `value` in der vorherigen `AUTHORIZER`-Eingabe auf `patchOperations` fest.

API-Schlüsseldateiformat in API Gateway

API Gateway kann API-Schlüssel aus externen Dateien im CSV-Format importieren und die Schlüssel dann einem oder mehreren Nutzungsplänen zuordnen. Die importierte Datei muss die Spalten `Name` und `Key` aufweisen. Bei Spaltenüberschriften wird z. B. die Groß-/Kleinschreibung nicht beachtet und die Spaltenreihenfolge ist beliebig:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

Ein `Key`-Wert muss zwischen 20 und 128 Zeichen lang sein. Ein `Name`-Wert darf nicht mehr als 1024 Zeichen enthalten.

Eine API-Schlüsseldatei kann z. B. auch die Spalten `Description`, `Enabled` oder `UsagePlanIds` aufweisen:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Wenn ein Schlüssel mit mehr als einem Nutzungsplan verknüpft ist, ist der `UsagePlanIds` Wert eine durch Kommas getrennte Zeichenfolge des Nutzungsplans IDs, die in doppelte oder einfache Anführungszeichen eingeschlossen ist, wie im folgenden Beispiel gezeigt:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsl"
```

Nicht erkannte Spalten sind zulässig, werden aber ignoriert. Der Standardwert ist eine leere Zeichenfolge oder ein boolescher `true`-Wert.

Derselbe API-Schlüssel kann mehrmals importiert werden, dabei überschreibt die neueste Version die vorherige. Zwei API-Schlüssel sind identisch, wenn sie den gleichen key-Wert haben.

 Note

Zu berücksichtigende bewährte Methoden finden Sie unter [the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”](#).

API-Schlüssel für REST APIs in API Gateway einrichten

Um API-Schlüssel einzurichten, gehen Sie wie folgt vor:

- Konfigurieren Sie API-Methoden, um einen API-Schlüssel erforderlich zu machen.
- Erstellen oder importieren Sie einen API-Schlüssel für die API in einer Region.

Vor der Einrichtung von API-Schlüsseln müssen Sie eine API erstellt haben und diese in einer Stufe bereitgestellt haben. Nach der Erstellung eines API-Schlüsselwerts kann dieser nicht geändert werden.

Anweisungen zum Erstellen und Bereitstellen einer API mithilfe der API Gateway-Konsole finden Sie unter [Entwickeln einer REST-API in API Gateway](#) und [REST-APIs in API Gateway bereitstellen](#).

Nach dem Erstellen eines API-Schlüssels muss dieser mit einem Nutzungsplan verknüpft werden. Weitere Informationen finden Sie unter [Nutzungspläne für REST APIs in API Gateway einrichten](#).

 Note

Zu berücksichtigende bewährte Methoden finden Sie unter [the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”](#).

Themen

- [Festlegen der Erforderlichkeit eines API-Schlüssels für eine Methode](#)
- [Erstellen Sie einen API-Schlüssel](#)
- [Importieren von API-Schlüsseln](#)

Festlegen der Erforderlichkeit eines API-Schlüssels für eine Methode

Im folgenden Verfahren wird beschrieben, wie Sie festlegen, dass für eine API-Methode ein API-Schlüssel erforderlich ist.

AWS-Managementkonsole

So konfigurieren Sie, dass für eine API-Methode ein API-Schlüssel erforderlich ist

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Resources (Ressourcen).
4. Erstellen Sie unter Resources eine neue Methode oder wählen Sie eine vorhandene aus.
5. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.

The screenshot shows the AWS API Gateway console interface. On the left, a navigation pane displays a tree structure with the following items: a 'Create resource' button, a root '/' resource with a 'GET' method, a '/pets' resource with a 'GET' method (highlighted), and an 'OPTIONS' button. Below the tree, there are links for 'POST', '/{petId}', 'GET', and 'OPTIONS'. The main panel on the right is titled 'Method request settings' and has an 'Edit' button in the top right corner, which is highlighted with a red box. The settings are organized into sections: 'Authorization' (NONE), 'Request validator' (None), 'API key required' (False), and 'SDK operation name' (Generated based on method and path). Below these are three expandable sections: 'Request paths (0)', 'URL query string parameters (2)', and 'HTTP request headers (2)'. The 'Request paths' section shows 'No request paths defined'. The 'URL query string parameters' section shows two parameters: 'page' and 'type', both with 'Required' set to 'False' and 'Caching' set to 'Inactive'. The 'HTTP request headers' section shows one header: 'header1', with 'Required' set to 'False' and 'Caching' set to 'Inactive'.

6. Wählen Sie API-Schlüssel erforderlich aus.
7. Wählen Sie Speichern.
8. Stellen Sie die API bereit (oder erneut bereit), damit die Änderung wirksam wird.

Wenn für die Option API-Schlüssel erforderlich der Wert `false` festgelegt ist und Sie die vorherigen Schritte nicht ausführen, wird keiner der API-Schlüssel, die einer API-Stufe zugeordnet sind, für diese Methode verwendet.

AWS CLI

Der folgende [put-method](#)-Befehl erstellt eine PUT-Methode, die einen API-Schlüssel erfordert:

```
aws apigateway put-method \  
  --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --authorization-type "NONE" \  
  --api-key-required
```

Der folgende [update-method](#)-Befehl aktualisiert eine vorhandene Methode, sodass ein API-Schlüssel erforderlich ist:

```
aws apigateway update-method \  
  --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --patch-operations op="replace",path="/apiKeyRequired",value="true"
```

REST API

Führen Sie einen der folgenden Schritte aus, damit für eine Methode ein API-Schlüssel erforderlich ist:

- Rufen Sie [method:put](#) auf, um eine Methode zu erstellen. Legen Sie `apiKeyRequired` in der Anforderungsnutzlast auf `true` fest.
- Rufen Sie [method:update](#) auf, um `apiKeyRequired` auf `true` zu setzen.

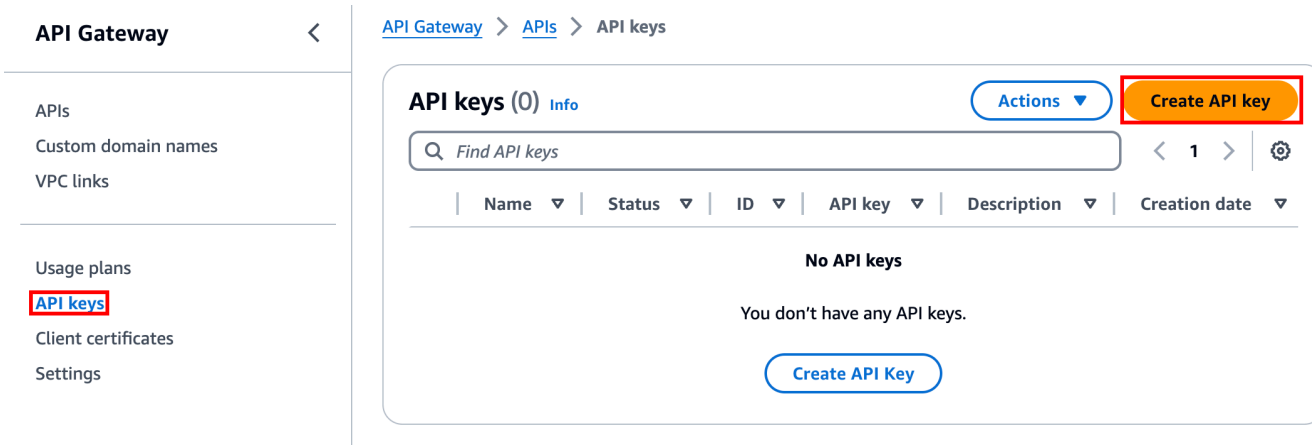
Erstellen Sie einen API-Schlüssel

Im folgenden Verfahren sehen Sie, wie Sie einen privaten API-Schlüssel erstellen. Wenn Sie Ihren API-Schlüssel importieren möchten, überspringen Sie diesen Schritt.

AWS-Managementkonsole

So erstellen Sie einen API-Schlüssel

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich von API Gateway API-Schlüssel.
4. Klicken Sie auf API-Schlüssel erstellen.



5. Geben Sie unter Name einen Namen ein.
6. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
7. Wählen Sie für API-Schlüssel die Option Automatisch generieren, damit API Gateway den Schlüsselwert generiert, oder wählen Sie Benutzerdefiniert, um Ihren eigenen Schlüsselwert zu erstellen.
8. Wählen Sie Speichern.

AWS CLI

Der folgende [create-api-key](#) Befehl erstellt einen API-Schlüssel:

```
aws apigateway create-api-key \
```

```
--name 'Dev API key' \  
--description 'API key for Devs' \  
--enabled
```

REST API

Rufen Sie [apikey:create](#) auf, um einen API-Schlüssel zu erstellen.

Importieren von API-Schlüsseln

Das folgende Verfahren beschreibt, wie Sie API-Schlüssel importieren. Wenn Sie bereits einen API-Schlüssel erstellt haben, überspringen Sie diesen Schritt.

AWS-Managementkonsole

So importieren Sie API-Schlüssel

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich API-Schlüssel.
4. Wählen das im Dropdown-Menü Aktionen und dann die Option API-Schlüssel importieren.
5. Wählen Sie zum Laden einer CSV-Datei Datei auswählen aus. Sie können die Schlüssel auch im Texteditor eingeben. Weitere Informationen zum Dateiformat finden Sie unter [the section called “API-Schlüsseldateiformat in API Gateway”](#).
6. Wählen Sie Fehler bei Warnungen aus, damit der Import im Falle eines Fehlers angehalten wird, oder wählen Sie Warnungen ignorieren aus, damit gültige Schlüssel auch bei einem auftretenden Fehler importiert werden.
7. Wählen Sie Import, um Ihre API-Schlüssel zu importieren.

AWS CLI

Der folgende [import-api-keys](#) Befehl importiert einen API-Schlüssel:

```
aws apigateway import-api-key \  
a--body fileb://keys.csv \  
--format csv
```


REST API

Rufen Sie [apikey:import](#) auf, um einen API-Schlüssel aus einer Datei zu importieren. Weitere Informationen zum Dateiformat finden Sie unter [API-Schlüsseldateiformat in API Gateway](#).

Sie können den Wert des neuen API-Schlüssels nicht ändern. Nachdem Sie Ihre API erstellt haben, konfigurieren Sie einen Nutzungsplan. Weitere Informationen finden Sie unter [Nutzungspläne für REST APIs in API Gateway einrichten](#).

Nutzungspläne für REST APIs in API Gateway einrichten

Bevor Sie einen Nutzungsplan erstellen, stellen Sie sicher, dass Sie Ihre API-Schlüssel eingerichtet haben. Weitere Informationen finden Sie unter [API-Schlüssel für REST APIs in API Gateway einrichten](#).

Themen

- [Migrieren Ihrer API auf Standard-Nutzungspläne \(falls erforderlich\)](#)
- [Erstellen eines Nutzungsplans](#)
- [Hinzufügen einer Stufe zu einem Nutzungsplan](#)
- [Hinzufügen eines API-Schlüssels zu einem Nutzungsplan](#)

Migrieren Ihrer API auf Standard-Nutzungspläne (falls erforderlich)

Wenn Sie API Gateway nach Einführung der Nutzungsplanfunktionalität (11. August 2016) verwenden, sind für Sie automatisch Nutzungspläne in allen unterstützten Regionen aktiviert.

Wenn Sie API Gateway vor diesem Datum eingeführt haben, müssen Sie möglicherweise auf Standardnutzungspläne migrieren. Die Option Nutzungspläne aktivieren wird angezeigt, bevor Sie zum ersten Mal Nutzungspläne in der ausgewählten Region verwenden. Durch die Aktivierung dieser Option werden Standardnutzungspläne für jede einzelne API-Stufe erstellt, der vorhandene API-Schlüssel zugeordnet sind. Im Standard-Nutzungsplan sind anfänglich keine Drossel- oder Kontingentlimits festgelegt, und die Zuordnungen zwischen den API-Schlüsseln und API-Stufen werden in die Nutzungspläne kopiert. Das API-Verhalten bleibt unverändert. Sie müssen jedoch die [UsagePlan](#)apiStagesEigenschaft verwenden, um die angegebenen API-Stufenwerte (apiIdundstage) den enthaltenen API-Schlüsseln (via [UsagePlanKey](#)) zuzuordnen, anstatt die [ApiKey](#)stageKeysEigenschaft zu verwenden.

Um zu überprüfen, ob Sie bereits auf Standardnutzungspläne migriert haben, führen Sie den CLI-Befehl [get-account](#) aus. In der Befehlsausgabe enthält die `features`-Liste den Eintrag `"UsagePlans"`, wenn Nutzungspläne aktiviert sind.

Sie können Ihre Nutzungspläne auch AWS CLI wie folgt APIs zu den Standard-Nutzungsplänen migrieren:

Um zu Standard-Nutzungsplänen zu migrieren, verwenden Sie AWS CLI

1. Rufen Sie diesen CLI-Befehl auf: [update-account](#).
2. Verwenden Sie für den `cli-input-json`-Parameter das folgende JSON-Konstrukt:

```
[
  {
    "op": "add",
    "path": "/features",
    "value": "UsagePlans"
  }
]
```

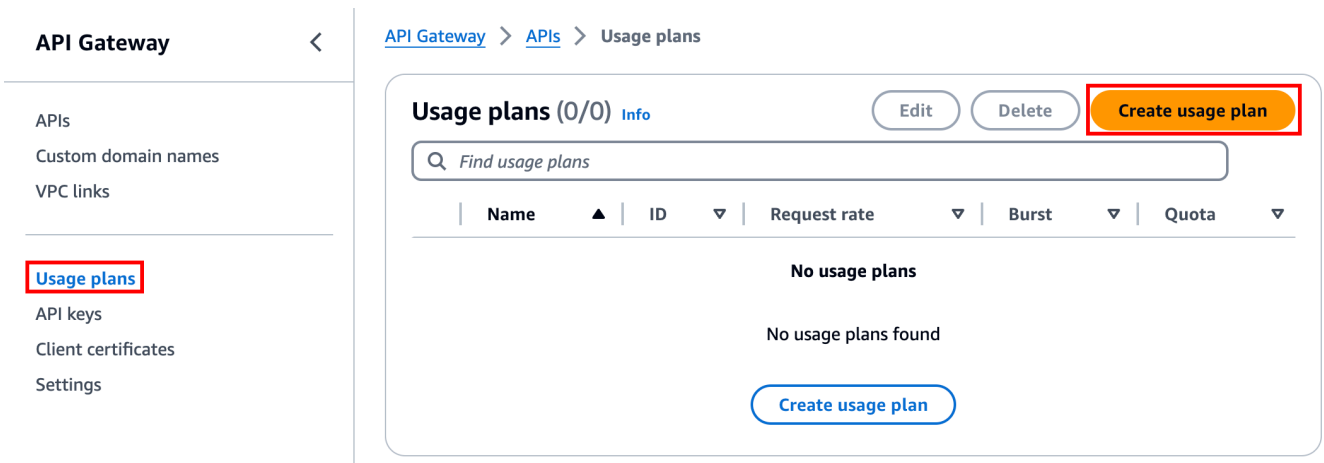
Erstellen eines Nutzungsplans

Im folgenden Verfahren wird das Erstellen eines Nutzungsplans beschrieben.

AWS-Managementkonsole

So erstellen Sie einen Nutzungsplan

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich von API Gateway Nutzungspläne und dann Nutzungspläne erstellen.



3. Geben Sie unter Name einen Namen ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Standardmäßig ermöglichen Nutzungspläne die Drosselung. Geben Sie eine Rate und einen Burst-Wert für Ihren Nutzungsplan ein. Wählen Sie Drosselung, um die Drosselung auszuschalten.
6. Standardmäßig ermöglichen Nutzungspläne ein Kontingent für einen bestimmten Zeitraum. Geben Sie unter Anfragen die Gesamtzahl der Anfragen ein, die ein Benutzer im Zeitraum Ihres Nutzungsplans stellen kann. Wählen Sie Kontingent, um das Kontingent zu deaktivieren.
7. Wählen Sie Nutzungsplan erstellen.

AWS CLI

Mit dem folgenden [create-usage-plan](#) Befehl wird ein Nutzungsplan erstellt, der zu Beginn des Monats zurückgesetzt wird:

```
aws apigateway create-usage-plan \
  --name "New Usage Plan" \
  --description "A new usage plan" \
  --throttle burstLimit=10,rateLimit=5 \
  --quota limit=500,offset=0,period=MONTH
```

REST API

Rufen Sie [usageplan:create](#) auf, um einen Nutzungsplan zu erstellen.

Hinzufügen einer Stufe zu einem Nutzungsplan

Das folgende Verfahren beschreibt, wie Sie eine Stufe zu einem Nutzungsplan hinzufügen.

AWS-Managementkonsole

So fügen Sie dem Nutzungsplan eine Stufe hinzu

1. Wählen Sie Ihren Nutzungsplan aus.
2. Wählen Sie auf der Registerkarte Zugeordnete Stufen die Option Stufe hinzufügen aus.

[API Gateway](#) > [APIs](#) > [Usage plans](#) > [MyUsagePlan](#)

MyUsagePlan

[Actions](#) ▼[Export usage data](#)

Usage plan details

Usage plan ID
abc123

Description
My new usage plan

AWS Marketplace product code
-

Rate
100 requests per second

Burst
20 requests

Quota
10 requests per month

[Associated stages](#)[Associated API keys](#)[Tags](#)

Associated stages (0) [Info](#)

[Edit](#)[Remove](#)[Add stage](#)

API ▼

Stage ▼

Method throttling

No stages

You don't have any stages.

[Add API stage](#)

3. Wählen Sie für API eine API aus.
4. Wählen Sie für Stufe eine Stufe aus.

5. (Optional) Gehen Sie wie folgt vor, um die Drosselung auf Methodenebene zu aktivieren:
 - a. Wählen Sie Drosselung auf Methodenebene und anschließend Methode hinzufügen aus.
 - b. Wählen Sie unter Ressource eine Ressource aus Ihrer API aus.
 - c. Wählen Sie unter Methode eine Methode aus Ihrer API aus.
 - d. Geben Sie eine Rate und einen Burst-Wert für Ihren Nutzungsplan ein.
6. Wählen Sie Zu Nutzungsplan hinzufügen aus.

AWS CLI

Der folgende [update-usage-plan](#) Befehl fügt einem Nutzungsplan die Prod Phase einer API hinzu:

```
aws apigateway update-usage-plan \  
  --usage-plan-id abc123 \  
  --patch-operations op="add",path="/apiStages",value="a1b1c2:Prod"
```

REST API

Rufen Sie [usageplan:update](#) auf, um einen Nutzungsplan zu aktualisieren.

Hinzufügen eines API-Schlüssels zu einem Nutzungsplan

Das folgende Verfahren zeigt, wie Sie einen API-Schlüssel zu einem Nutzungsplan hinzufügen.

AWS-Managementkonsole

So fügen Sie dem Nutzungsplan einen Schlüssel hinzu

1. Wählen Sie auf der Registerkarte Zugeordnete API-Schlüssel die Option API-Schlüssel hinzufügen aus.

MyUsagePlan

Actions ▼

Export usage data

Usage plan details

Usage plan ID

abc123

Description

My new usage plan

AWS Marketplace product code

-

Rate

100 requests per second

Burst

20 requests

Quota

10 requests per month

Associated stages

Associated API keys

Tags

API keys (0) [Info](#)

Actions ▼

Add API key

< 1 > ⚙

Name ▼

Status ▼

ID ▼

API key ▼

Requests remaining this month ▼

No API keys.

This usage plan has no API keys.

Add API key

2. a. Um Ihrem Nutzungsplan einen vorhandenen Schlüssel zuzuordnen, wählen Sie Bestehenden Schlüssel hinzufügen und wählen Sie dann Ihren vorhandenen Schlüssel aus dem Drop-down-Menü aus.
- b. Um einen neuen API-Schlüssel zu erstellen, wählen Sie Neuen Schlüssel erstellen und hinzufügen aus und erstellen Sie dann einen neuen Schlüssel. Weitere Informationen zum Erstellen eines neuen Schlüssels finden Sie unter [Erstellen Sie einen API-Schlüssel](#).
3. Wählen Sie API-Schlüssel hinzufügen.

AWS CLI

Der folgende [create-usage-plan-key](#) Befehl ordnet einen vorhandenen API-Schlüssel einem Nutzungsplan zu:

```
aws apigateway create-usage-plan-key \  
  --usage-plan-id a1b2c3 \  
  --key-type "API_KEY" \  
  --key-id aaa111bbb
```

REST API

Rufen Sie [usageplankey:create](#) auf, um einen vorhandenen API-Schlüssel mit einem Nutzungsplan zu verknüpfen.

Sie können Ihre APIs Schlüssel auch direkt einem Nutzungsplan zuordnen, wenn Sie sie importieren. Rufen Sie [apikey:import](#) auf und fügen Sie einen oder mehrere API-Schlüssel direkt zum angegebenen Nutzungsplan hinzu. Die Anforderungsnutzlast sollte API-Schlüsselwerte, die zugehörige Nutzungsplan-ID, boolesche Flags (um anzugeben, dass die Schlüssel für den Nutzungsplan aktiviert sind) und ggf. Namen und Beschreibungen der API-Schlüssel enthalten.

Im folgenden Beispiel der `apikey:import`-Anforderung werden drei API-Schlüssel (identifiziert durch `key`, `name` und `description`) zu einem Nutzungsplan (identifiziert durch `usageplanIds`) hinzugefügt:

```
POST /apikeys?mode=import&format=csv&failonwarnings=false HTTP/1.1  
Host: apigateway.us-east-1.amazonaws.com  
Content-Type: text/csv  
Authorization: ...  
  
key,name, description, enabled, usageplanIds  
abcdef1234ghijklmnop8901234567, importedKey_1, firstone, tRuE, n371pt  
abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt  
abcdef1234ghijklmnop9012345678, importedKey_3, , true, n371pt
```

Als Ergebnis werden drei `UsagePlanKey`-Ressourcen erstellt und zu `UsagePlan` hinzugefügt.

Auf diese Weise können Sie auch API-Schlüssel zu mehreren Nutzungsplänen hinzufügen. Dazu ändern Sie jeden `usageplanIds`-Spaltenwert in eine CSV-Zeichenfolge, die die ausgewählten Nutzungsplan-IDs in Anführungszeichen angibt ("`n371pt,m282qs`" oder '`n371pt,m282qs`').

Note

Ein API-Schlüssel kann mehreren Nutzungsplänen zugeordnet werden. Ein Nutzungsplan kann mehreren Stufen zugeordnet werden. Ein bestimmter API-Schlüssel kann jedoch nur einem Nutzungsplan für jede Stufe Ihres API zugeordnet werden.

Pflegen Sie einen Nutzungsplan für REST APIs in API Gateway

Im Rahmen der Nutzungsplanverwaltung werden verwendete und verbleibende Kontingente in einem bestimmten Zeitraum überwacht und die verbleibenden Kontingente gegebenenfalls um eine angegebene Menge erhöht. In den folgenden Verfahren wird beschrieben, wie Sie Kontingente überwachen.

AWS-Managementkonsole

So überwachen Sie verwendete und verbleibende Kontingente

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Nutzungspläne.
3. Wählen Sie einen Nutzungsplan aus.
4. Wählen Sie die Registerkarte Zugeordnete API-Schlüssel, um die Anzahl der für den Zeitraum verbleibenden Anfragen für jeden Schlüssel anzuzeigen.
5. (Optional) Wählen Sie Nutzungsdaten exportieren und dann ein Startdatum und ein Enddatum aus. Wählen Sie dann JSON oder CSV als Exportdatenformat und anschließend Exportieren aus.

Im folgenden Beispiel wird eine exportierte Datei angezeigt.

```
{
  "px1KW6...qBaz0JH": [
    0,
    5000
  ],
  [
    0,
    5000
  ]
}
```



```
    ],  
    [  
      0,  
      10  
    ]  
  ]  
}
```

Die Nutzungsdaten im Beispiel geben die tägliche Nutzung eines API-Clients, der vom API-Schlüssel (px1KW6 . . . qBaz0JH) identifiziert wird, zwischen dem 1. August 2016 und dem 3. August 2016 an. In den täglichen Nutzungsdaten werden jeweils die verwendeten und die verbleibenden Kontingente angezeigt. In diesem Beispiel hat der Abonnent die ihm zugeordneten Kontingente gar nicht verwendet und der API-Besitzer oder Administrator hat das verbleibende Kontingent für den dritten Tag von 5.000 auf 10 gesenkt.

In den folgenden Verfahren wird beschrieben, wie Sie Kontingente ändern.

So erhöhen Sie verbleibende Kontingente

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Nutzungspläne.
3. Wählen Sie einen Nutzungsplan aus.
4. Wählen Sie die Registerkarte Zugeordnete API-Schlüssel, um die Anzahl der für den Zeitraum verbleibenden Anfragen für jeden Schlüssel anzuzeigen.
5. Wählen Sie einen API-Schlüssel und dann Nutzungserweiterung gewähren aus.
6. Geben Sie eine Zahl für das Kontingent Verbleibende Anforderungen an. Sie können die Anzahl der Umbenennungsanfragen erhöhen oder die Anzahl der verbleibenden Anfragen für den Zeitraum Ihres Nutzungsplans verringern.
7. Wählen Sie Kontingent aktualisieren aus.

AWS CLI

In den folgenden [update-usage-plan](#)Beispielen werden die Drosselungseinstellungen auf Methodenebene in einem Nutzungsplan hinzugefügt, entfernt oder geändert.

 Note

Achten Sie darauf, `us-east-1` auf den korrekten Regionswert für Ihre API zu ändern.

Um ein Ratenlimit zur Drosselung einer individuellen Ressource und Methode hinzuzufügen oder zu ersetzen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="replace",path="/apiStages/apiId:stage/throttle/resourcePath/httpMethod/rateLimit",value="0.1"
```

Um ein Burst-Limit zur Drosselung einer individuellen Ressource und Methode hinzuzufügen oder zu ersetzen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="replace",path="/apiStages/apiId:stage/throttle/resourcePath/httpMethod/burstLimit",value="1"
```

Um die Drosseleinstellungen auf Methodenebene für eine individuelle Ressource und Methode zu entfernen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="remove",path="/apiStages/apiId:stage/throttle/resourcePath/httpMethod",value=""
```

Um alle Drosseleinstellungen einer API auf Methodenebene zu entfernen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="remove",path="/apiStages/apiId:stage/throttle ",value=""
```

Hier sehen Sie ein Beispiel unter Verwendung der Pet Store-Beispiel-API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="replace",path="/apiStages/apiId:stage/throttle",value="{\"/pets/GET\":{\"rateLimit\":1.0,\"burstLimit\":1},\"//GET\":{\"rateLimit\":1.0,\"burstLimit\":1}}\""
```

REST API

Rufen Sie [usageplan:update](#) auf, um einen Nutzungsplan zu verwalten.

Erstellen und konfigurieren Sie API-Schlüssel und Nutzungspläne mit CloudFormation

Sie können verwenden CloudFormation , um API-Schlüssel für API-Methoden vorzuschreiben und einen Nutzungsplan für eine API zu erstellen. Die CloudFormation Beispielvorlage macht Folgendes:

- Mit den Methoden GET und POST wird eine API-Gateway-API erstellt.
- Erfordert einen API-Schlüssel für die Methoden GET und POST. Diese API ruft Schlüssel aus dem X-API-KEY-Header jeder eingehenden Anfrage ab.
- Erstellt einen API-Schlüssel
- Es wird ein Nutzungsplan erstellt, um ein monatliches Kontingent von 1 000 Anfragen pro Monat, ein Limit bei der Drosselungsrate von 100 Anfragen pro Sekunde und ein Drosselungs-Burst-Limit von 200 Anfragen pro Sekunde festzulegen.
- Gibt für die GET-Methode ein Limit von 50 Anfragen pro Sekunde für die Drosselungsrate auf Methodenebene und ein Burst-Limit für die Drosselung auf Methodenebene von 100 Anfragen pro Sekunde an.
- Ordnet dem Nutzungsplan API-Stufen und API-Schlüssel zu

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
  KeyName:
    Type: String
    Default: MyKeyName
    Description: Name of an API key
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: keys-api
      ApiKeySourceType: HEADER
  PetsResource:
```

```
Type: 'AWS::ApiGateway::Resource'
Properties:
  RestApiId: !Ref Api
  ParentId: !GetAtt Api.RootResourceId
  PathPart: 'pets'
PetsMethodGet:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref PetsResource
    HttpMethod: GET
    ApiKeyRequired: true
    AuthorizationType: NONE
    Integration:
      Type: HTTP_PROXY
      IntegrationHttpMethod: GET
      Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
PetsMethodPost:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    RestApiId: !Ref Api
    ResourceId: !Ref PetsResource
    HttpMethod: POST
    ApiKeyRequired: true
    AuthorizationType: NONE
    Integration:
      Type: HTTP_PROXY
      IntegrationHttpMethod: GET
      Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
ApiDeployment:
  Type: 'AWS::ApiGateway::Deployment'
  DependsOn:
    - PetsMethodGet
  Properties:
    RestApiId: !Ref Api
    StageName: !Sub '${StageName}'
UsagePlan:
  Type: AWS::ApiGateway::UsagePlan
  DependsOn:
    - ApiDeployment
  Properties:
    Description: Example usage plan with a monthly quota of 1000 calls and method-
level throttling for /pets GET
    ApiStages:
```

```

    - ApiId: !Ref Api
      Stage: !Sub '${StageName}'
      Throttle:
        "/pets/GET":
          RateLimit: 50.0
          BurstLimit: 100
      Quota:
        Limit: 1000
        Period: MONTH
      Throttle:
        RateLimit: 100.0
        BurstLimit: 200
      UsagePlanName: "My Usage Plan"
  ApiKey:
    Type: AWS::ApiGateway::ApiKey
    Properties:
      Description: API Key
      Name: !Sub '${KeyName}'
      Enabled: True
  UsagePlanKey:
    Type: AWS::ApiGateway::UsagePlanKey
    Properties:
      KeyId: !Ref ApiKey
      KeyType: API_KEY
      UsagePlanId: !Ref UsagePlan
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'

```

Konfigurieren einer Methode zur Verwendung von API-Schlüsseln mit einer OpenAPI-Definition

Mithilfe einer OpenAPI-Definition können Sie vorgeben, dass API-Schlüssel für eine Methode verwendet werden müssen.

Erstellen Sie für jede Methode ein Sicherheitsanforderungsobjekt, das einen API-Schlüssel zum Aufrufen dieser Methode abfragt. Anschließend geben Sie `api_key` in der Sicherheitsdefinition vor. Nachdem Sie Ihre API erstellt haben fügen Sie die neue API-Stufe Ihrem Nutzungsplan hinzu.

Im folgenden Beispiel wird eine API erstellt und ein API-Schlüssel für die GET- und POST-Methoden abgefragt:

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "2024-03-14T20:20:12Z",
    "title" : "keys-api"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      },
      "post" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "securityDefinitions" : {
    "api_key" : {
      "type" : "apiKey",
      "name" : "x-api-key",
      "in" : "header"
    }
  }
}
```

```
}  
}
```

OpenAPI 3.0

```
{  
  "openapi" : "3.0.1",  
  "info" : {  
    "title" : "keys-api",  
    "version" : "2024-03-14T20:20:12Z"  
  },  
  "servers" : [ {  
    "url" : "{basePath}",  
    "variables" : {  
      "basePath" : {  
        "default" : "v1"  
      }  
    }  
  } ],  
  "paths" : {  
    "/pets" : {  
      "get" : {  
        "security" : [ {  
          "api_key" : [ ]  
        } ],  
        "x-amazon-apigateway-integration" : {  
          "httpMethod" : "GET",  
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",  
          "passthroughBehavior" : "when_no_match",  
          "type" : "http_proxy"  
        }  
      },  
      "post" : {  
        "security" : [ {  
          "api_key" : [ ]  
        } ],  
        "x-amazon-apigateway-integration" : {  
          "httpMethod" : "GET",  
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",  
          "passthroughBehavior" : "when_no_match",  
          "type" : "http_proxy"  
        }  
      }  
    }  
  }  
}
```

```

    }
  },
  "components" : {
    "securitySchemes" : {
      "api_key" : {
        "type" : "apiKey",
        "name" : "x-api-key",
        "in" : "header"
      }
    }
  }
}

```

Nutzungspläne für REST APIs in API Gateway testen

Verwenden wir als Beispiel die PetStore API, die in erstellt wurde [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#). Die API ist für die Verwendung des API-Schlüssels konfiguriert `Hiorr45VR...c4GJc`. In den folgenden Schritten wird das Testen eines Nutzungsplans beschrieben.

So testen Sie den Nutzungsplan

- Führen Sie eine GET-Anforderung für die Ressource "Pets" (`/pets`) mit den Abfrageparametern `?type=...&page=...` der API (zum Beispiel `xbvxlpijch`) in einem Nutzungsplan aus:

```

GET /testStage/pets?type=dog&page=1 HTTP/1.1
x-api-key: Hiorr45VR...c4GJc
Content-Type: application/x-www-form-urlencoded
Host: xbvxlpijch.execute-api.ap-southeast-1.amazonaws.com
X-Amz-Date: 20160803T001845Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160803/ap-southeast-1/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date;x-api-key, Signature={sigv4_hash}

```

Note

Sie müssen diese Anforderung an die Komponente `execute-api` von API Gateway übermitteln und den erforderlichen API-Schlüssel (zum Beispiel `Hiorr45VR...c4GJc`) im entsprechenden `x-api-key`-Header bereitstellen.

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit den angeforderten Ergebnissen aus dem Backend zurückgegeben. Falls Sie den `x-api-key`-Header nicht oder mit einem falschen Schlüssel gesetzt haben, erhalten Sie als Antwort die Meldung `403 Forbidden`. Wenn Sie allerdings nicht festgelegt haben, dass für die Methode ein API-Schlüssel erforderlich ist, erhalten Sie in jedem Fall die Antwort `200 OK` – unabhängig davon, ob der `x-api-key`-Header korrekt oder fehlerhaft eingerichtet wurde – und die Ablehnungs- und Kontingentlimits werden ignoriert.

Im Falle, dass API Gateway aufgrund eines internen Fehlers die im Nutzungsplan enthaltenen Drosselungs- und Kontingentlimits für diese Anforderung nicht erzwingen kann, führt API Gateway die Anforderung ohne Berücksichtigung der Limits aus. Es wird jedoch die Fehlermeldung `CloudWatch in Usage Plan check failed due to an internal error` protokolliert. Diese gelegentlich auftretenden Fehler können Sie ignorieren.

Aufrufen einer Methode mithilfe eines API-Schlüssels

Abhängig von dem von Ihnen gewählten API-Schlüsselquelltyp verwenden Sie eines der folgenden Verfahren für die Verwendung eines Header-bezogenen API-Schlüssels oder eines vom Genehmiger zurückgegebenen API-Schlüssels für einen Methodenaufruf:

Um Header-bezogene API-Schlüssel zu verwenden:

1. Erstellen Sie eine API mit den gewünschten API-Methoden und stellen Sie die API dann für eine Stufe bereit.
2. Erstellen Sie einen neuen Nutzungsplan oder wählen Sie einen bereits vorhandenen Plan aus. Fügen Sie die bereitgestellte API-Stufe dem Nutzungsplan hinzu. Ordnen Sie dem Nutzungsplan einen API-Schlüssel hinzu oder wählen Sie einen im Plan vorhandenen API-Schlüssel. Notieren Sie den ausgewählten API-Schlüsselwert.
3. Richten Sie API-Methoden ein, um einen API-Schlüssel zu fordern.
4. Stellen Sie die API derselben Stufe bereit. Wenn Sie die API einer neuen Stufe bereitstellen, stellen Sie sicher, dass Sie den Nutzungsplan aktualisieren, um ihn der neuen API-Stufe zuzuordnen.
5. Rufen Sie die API mithilfe des API-Schlüssels auf. Der folgende CURL-Beispielbefehl ruft mithilfe eines API-Schlüssels die GET-Methode für die `getUsers`-Ressource der `prod`-Stufe einer API auf.

```
curl -H "X-API-Key: abcd1234" 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Der Client kann nun die API-Methoden aufrufen, während er den `x-api-key`-Header mit dem gewählten API-Schlüssel als Header-Wert bereitstellt. Ein Aufruf könnte in diesem Fall wie folgt aussehen:

Um Genehmiger-bezogene API-Schlüssel zu verwenden:

1. Erstellen Sie eine API mit den gewünschten API-Methoden und stellen Sie die API dann für eine Stufe bereit.
2. Erstellen Sie einen neuen Nutzungsplan oder wählen Sie einen bereits vorhandenen Plan aus. Fügen Sie die bereitgestellte API-Stufe dem Nutzungsplan hinzu. Ordnen Sie dem Nutzungsplan einen API-Schlüssel hinzu oder wählen Sie einen im Plan vorhandenen API-Schlüssel. Notieren Sie den ausgewählten API-Schlüsselwert.
3. Erstellen eines token-basierten Lambda-Genehmigers. `usageIdentifierKey: {api-key}` als Root-Level-Eigenschaft der Autorisierungsantwort einschließen. Anweisungen zum Erstellen eines Token-basierten Genehmigers finden Sie unter [the section called “Beispiel für eine Lambda-Funktion für TOKEN-Genehmiger”](#).
4. Richten Sie API-Methoden ein, um einen API-Schlüssel zu fordern, und aktivieren Sie den Lambda-Genehmiger für die Methoden.
5. Stellen Sie die API derselben Stufe bereit. Wenn Sie die API einer neuen Stufe bereitstellen, stellen Sie sicher, dass Sie den Nutzungsplan aktualisieren, um ihn der neuen API-Stufe zuzuordnen.

Der Client kann nun die Methoden aufrufen, für die ein API-Schlüssel gefordert wird, ohne explizit einen API-Schlüssel bereitstellen zu müssen. Der vom Genehmiger zurückgegebene API-Schlüssel wird automatisch verwendet.

Dokumentation für REST APIs in API Gateway

Um Kunden dabei zu helfen, Ihre API zu verstehen und zu verwenden, sollten Sie die API dokumentieren. Dazu können Sie mithilfe von API Gateway Hilfeinhalte für einzelne API-Entitäten als festen Bestandteil Ihres API-Entwicklungsprozesses hinzufügen und aktualisieren. API Gateway speichert die Quellinhalte und ermöglicht es Ihnen, verschiedene Versionen der Dokumentation

zu archivieren. Sie können eine Dokumentationsversion mit einer API-Phase verknüpfen, einen Dokumentations-Snapshot zu einer bestimmten Phase in eine externe OpenAPI-Datei exportieren und die Datei zur Veröffentlichung der Dokumentation verteilen.

Um Ihre API zu dokumentieren, können Sie die [API Gateway-REST-API](#) aufrufen, eine der APIs verwenden [AWS SDKs](#), die [AWS CLI](#) für API Gateway verwenden oder die API Gateway Gateway-Konsole verwenden. Sie haben auch die Möglichkeit, die in einer externen OpenAPI-Datei definierten Dokumentationsbausteine zu importieren oder zu exportieren.

Wenn Sie API-Dokumentation mit Entwicklern teilen möchten, können Sie ein Entwicklerportal verwenden. Ein Beispiel finden Sie unter [Integration ReadMe mit API Gateway, um Ihren Developer Hub auf dem neuesten Stand zu halten](#), oder im Blog [So optimieren Sie die API-Entwicklung auf Amazon API Gateway SmartBear SwaggerHub Using](#) im AWS Partner Network (APN).

Themen

- [Darstellung der API-Dokumentation in API Gateway](#)
- [Dokumentieren einer API mit der API Gateway-Konsole](#)
- [Veröffentlichen der API-Dokumentation mithilfe der API Gateway-Konsole](#)
- [Dokumentieren einer API mit der API Gateway-REST-API](#)
- [Veröffentlichen der API-Dokumentation mithilfe der API Gateway-REST-API](#)
- [Importieren einer API-Dokumentation](#)
- [Zugriff auf API-Dokumentation in API Gateway steuern](#)

Darstellung der API-Dokumentation in API Gateway

Eine API Gateway-API-Dokumentation besteht aus einzelnen Bestandteilen (Bausteinen), die sich auf bestimmte API-Entitäten beziehen, z. B. API, Ressource, Methode, Anforderung, Antwort, Nachrichtenparameter (z. B. Pfad, Abfrage, Header) sowie Genehmiger und Modelle.

In API Gateway wird ein Dokumentationsteil durch eine [DocumentationPart](#)-Ressource dargestellt. Die API-Dokumentation als Ganzes wird durch die [DocumentationParts](#)-Sammlung repräsentiert.

Die Dokumentation einer API umfasst das Erstellen von `DocumentationPart`-Instances, das Hinzufügen dieser Instances zur `DocumentationParts`-Sammlung und die Pflege der verschiedenen Versionen der Dokumentationsbausteine, die im API-Entwicklungsprozess entstehen.

Themen

- [Dokumentationsbausteine](#)
- [Dokumentationsversionen](#)

Dokumentationsbausteine

Eine [DocumentationPart](#)-Ressource ist ein JSON-Objekt, das den Dokumentationsinhalt speichert, der für eine einzelne API-Entität gilt. Dazu gehören alle UTF-8-Inhalte und alle wichtigen Lokalisierungssprachen für Ihre Dokumentation. Das Feld `properties` enthält den Dokumentationsinhalt in Form einer Übersicht von Schlüssel-Wert-Paaren. Die Eigenschaft `location` identifiziert die zugehörige API-Entität.

Sie, der API-Entwickler, bestimmen die Form einer Content Map. Der Wert eines Schlüssel-Wert-Paars kann eine Zeichenfolge, eine Zahl, ein boolescher Wert, ein Objekt oder ein Array sein. Die Form des `location`-Objekts hängt vom jeweiligen Entitätstyp ab.

Die `DocumentationPart`-Ressource unterstützt das Vererben von Inhalten, das heißt, der Dokumentationsinhalt einer API-Entität gilt auch für die untergeordneten Elemente dieser API-Entität. Weitere Informationen zur Definition untergeordneter Entitäten und zur Vererbung von Inhalten finden Sie unter [Inherit Content from an API Entity of More General Specification](#).

Lokation eines Dokumentationsbausteins

Die [Location-Eigenschaft](#) einer [DocumentationPart](#)-Instanz identifiziert eine API-Entität, für die der zugehörige Inhalt gilt. Die API-Entität kann eine API-Gateway-REST-API-Ressource sein [RestApi](#), z. B. [Resource](#), [Method](#) [MethodResponse](#), [Authorizer](#) oder [Model](#). Die Entität kann auch ein Nachrichtenparameter sein, z. B. ein URL-Pfadparameter, ein Abfragezeichenfolge-Parameter, ein Anforderungs- oder Antwort-Header-Parameter, ein Anforderungs- oder Antworttext oder ein Antwortstatuscode.

Um eine API-Entität anzugeben, wählen Sie für das Attribut `type` des Objekts `location` eine der folgenden Optionen: `API`, `AUTHORIZER`, `MODEL`, `RESOURCE`, `METHOD`, `PATH_PARAMETER`, `QUERY_PARAMETER`, `REQUEST_HEADER`, `REQUEST_BODY`, `RESPONSE`, `RESPONSE_HEADER` oder `RESPONSE_BODY`.

Je nach `type` einer API-Entität können Sie auch andere `location`-Attribute angeben, z. B. [method](#), [name](#), [path](#) und [statusCode](#). Nicht alle diese Attribute gelten für eine bestimmte API-Entität. Zum Beispiel sind `type`, `path`, `name` und `statusCode` gültige Attribute der Entität `RESPONSE`; nur `type` und `path` sind gültige Lokationsattribute der Entität `RESOURCE`. Wenn Sie ein ungültiges Feld in der

location eines `DocumentationPart` für eine bestimmte API-Entität angeben, führt dies zu einem Fehler.

Nicht alle gültigen `location`-Felder sind erforderlich. Beispielsweise ist `type` ein gültiges und erforderliches `location`-Feld für alle API-Entitäten. Die Attribute `method`, `path` und `statusCode` hingegen sind gültige, aber keine erforderlichen Attribute für die Entität `RESPONSE`. Sofern nicht explizit angegeben, übernimmt ein gültiges `location`-Feld seinen Standardwert. Der Standardwert für `path` ist `/`, d. h. die Stammressource einer API. Der Standardwert für `method` oder `statusCode` ist `*`, d. h. jeder beliebige Methoden- bzw. Statuscodewert.

Inhalt eines Dokumentationsbausteins

Der Wert `properties` ist als JSON-Zeichenfolge kodiert. Der Wert `properties` enthält alle Informationen, die Sie auswählen, um Ihre Dokumentationsanforderungen zu erfüllen. Das folgende Beispiel zeigt eine gültige Content Map:

```
{
  "info": {
    "description": "My first API with Amazon API Gateway."
  },
  "x-custom-info" : "My custom info, recognized by OpenAPI.",
  "my-info" : "My custom info not recognized by OpenAPI."
}
```

Auch wenn API Gateway jede gültige JSON-Zeichenfolge als Content Map akzeptiert, werden die Inhaltsattribute als zwei Kategorien behandelt: als Attribute, die von OpenAPI erkannt werden, und als Attribute, die nicht von OpenAPI erkannt werden. Im vorherigen Beispiel werden `info`, `description` und `x-custom-info` von OpenAPI als OpenAPI-Standardobjekt, -eigenschaft oder -erweiterung erkannt. Im Gegensatz dazu ist `my-info` nicht mit der OpenAPI-Spezifikation kompatibel. API Gateway verteilt OpenAPI-kompatible Inhaltsattribute in die API-Entitätsdefinitionen von den verknüpften `DocumentationPart`-Instances. Die nicht kompatiblen Inhaltsattribute werden von API Gateway nicht in die API-Entitätsdefinitionen verteilt.

Ein weiteres Beispiel, bei dem `DocumentationPart` für eine `Resource`-Entität gilt:

```
{
  "location" : {
    "type" : "RESOURCE",
    "path": "/pets"
  },
}
```

```
"properties" : {  
  "summary" : "The /pets resource represents a collection of pets in PetStore.",  
  "description": "... a child resource under the root...",  
}  
}
```

Hier sind sowohl `type` als auch `path` gültige Felder zur Identifizierung des Ziels des Typs `RESOURCE`. Für die Stammressource (/) können Sie das Feld `path` weglassen.

```
{  
  "location" : {  
    "type" : "RESOURCE"  
  },  
  "properties" : {  
    "description" : "The root resource with the default path specification."  
  }  
}
```

Dies entspricht der folgenden `DocumentationPart`-Instance:

```
{  
  "location" : {  
    "type" : "RESOURCE",  
    "path": "/"  
  },  
  "properties" : {  
    "description" : "The root resource with an explicit path specification"  
  }  
}
```

Vererben von Inhalten aus einer API-Entität allgemeinerer Spezifikationen

Der Standardwert eines optionalen `location`-Felds bietet eine musterhafte Beschreibung einer API-Entität. Unter Verwendung des Standardwerts im Objekt `location` können Sie eine allgemeine Beschreibung in der `properties`-Map zu einer `DocumentationPart`-Instance mit dieser Art von `location`-Muster hinzufügen. API Gateway extrahiert die entsprechenden OpenAPI-Dokumentationsattribute aus dem `DocumentationPart` der allgemeinen API-Entität und fügt sie in eine spezifische API-Entität mit den `location`-Feldern ein, die dem allgemeinen `location`-Muster entsprechen oder mit dem genauen Wert übereinstimmen, es sei denn, der betreffenden Entität ist

bereits eine `DocumentationPart`-Instance zugeordnet. Dieses Verhalten wird auch als Vererbung von Inhalten aus einer API-Entität allgemeinerer Spezifikationen bezeichnet.

Die Vererbung von Inhalten gilt für bestimmte API-Entitätentypen nicht. Weitere Details finden Sie in der nachfolgenden Tabelle.

Wenn eine API-Entität mit mehr als einem `DocumentationPart`-Lokationsmuster übereinstimmt, übernimmt die Entität den Dokumentationsbaustein mit den "location"-Feldern der höchsten Priorität und Spezifität. Die Rangfolge ist `path > statusCode`. Für den Abgleich mit dem `path`-Feld wählt API Gateway die Entität mit dem spezifischsten Pfadwert. In der folgenden Tabelle wird dies mit ein paar Beispielen veranschaulicht.

Fall	path	stat e	name	Anmer en
1	/pets	*	id	Die mit diesem Lokationsmuster verknüpften Dokumentation wird von den Entitäten übernommen, die mit dem Lokationsmuster übereinstimmen.

Fall	path	stat e	name	Anmer en	
2	/pets	200	id	Die mit diesem Lokations muster verknüpft e Dokumenta tion wird von den Entitäten übernomme n, die mit dem Lokations muster übereinst immen, sofern Fall 1 und 2 abgeglich en werden, da Fall 2	

Fall	path	stat e	name	Anmer en	
				spezifisc her ist als Fall 1.	

Fall	path	stat e	name	Anmer en	
3	/pets/ petId	*	id	Die mit diesem Lokations muster verknüpft e Dokumenta tion wird von den Entitäten übernomme n, die mit dem Lokations muster übereinst immen, wenn Fall 1, 2 und 3 abgeglich en werden, da Fall 3	

Fall	path	stat e	name	Anmer en	
				eine höhere Priorität hat als Fall 2 und spezifisc her ist als Fall 1.	

Hier ein weiteres Beispiel für eine allgemeinere `DocumentationPart`-Instance im Vergleich zu einer spezifischeren Instance. Die folgende allgemeine Fehlermeldung `"Invalid request error"` wird in die OpenAPI-Definitionen der 400-Fehlermeldungen eingefügt, sofern sie nicht überschrieben werden.

```
{
  "location" : {
    "type" : "RESPONSE",
    "statusCode": "400"
  },
  "properties" : {
    "description" : "Invalid request error."
  }
}
```

Mit der folgenden Überschreibung verfügen die 400-Antworten auf alle Methoden der `/pets-`Ressource stattdessen über die Beschreibung `"Invalid petId specified"`.

```
{
  "location" : {
```

```
    "type" : "RESPONSE",
    "path": "/pets",
    "statusCode": "400"
  },
  "properties" : "{
    "description" : "Invalid petId specified."
  }"
}
```

Zulässige "location"-Felder für **DocumentationPart**

Die folgende Tabelle zeigt die gültigen und erforderlichen Felder sowie die geltenden Standardwerte einer [DocumentationPart](#)Ressource, die einem bestimmten Typ von API-Entitäten zugeordnet ist.

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
API	<pre>{ "location": { "type": "API" }, ... }</pre>	type	–	Nein
Ressource	<pre>{ "location": { "type": "RESOURCE" }, "path": "<i>resource_path</i> " }, ... }</pre>	type	Der Standardwert von path ist /.	Nein
Methode	<pre>{ "location": { "type": "METHOD", </pre>	type	Die Standardwerte von path und method sind / bzw. *.	Ja, Abgleich von path nach Präfix und von method

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
	<pre> "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> " }, ... } </pre>			nach beliebigen Werten
Abfrageparameter	<pre> { "location": { "type": "QUERY_PARAMETER", "path": "<i>resource_path</i> ", "method": "<i>HTTP_verb</i> ", "name": "<i>query_parameter_name</i> " }, ... } </pre>	type	Die Standardwerte von path und method sind / bzw. *.	Ja, Abgleich von path nach Präfix und von method nach exakten Werten
Anforderungstext	<pre> { "location": { "type": "REQUEST_BODY", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> " }, ... } </pre>	type	Die Standardwerte von path und method sind / bzw. *.	Ja, Abgleich von path nach Präfix und von method nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbte Inhalte
Parameter des Anforderungs-Headers	<pre>{ "location": { "type": "REQUEST_HEADER", "path": "<i>resource_path</i> ", "method": "<i>HTTP_verb</i> ", "name": "<i>header_name</i> " }, ... }</pre>	type, name	Die Standardwerte von path und method sind / bzw. *.	Ja, Abgleich von path nach Präfix und von method nach exakten Werten
Pfadparameter der Anforderung	<pre>{ "location": { "type": "PATH_PARAMETER", "path": "<i>resource/{path_parameter_name}</i> ", "method": "<i>HTTP_verb</i> ", "name": "<i>path_parameter_name</i> " }, ... }</pre>	type, name	Die Standardwerte von path und method sind / bzw. *.	Ja, Abgleich von path nach Präfix und von method nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbte Inhalte
Antwort	<pre>{ "location": { "type": "RESPONSE", "path": "<i>resource_path</i>", "method": "<i>http_verb</i>", "statusCode": "<i>status_code</i>" }, ... }</pre>	type	Die Standardwerte von path, method und statusCode sind /, * bzw. *.	Ja, Abgleich von path nach Präfix und von method und statusCode nach exakten Werten
Antwort-Header	<pre>{ "location": { "type": "RESPONSE_HEADER", "path": "<i>resource_path</i>", "method": "<i>http_verb</i>", "statusCode": "<i>status_code</i>", "name": "<i>header_name</i>" }, ... }</pre>	type, name	Die Standardwerte von path, method und statusCode sind /, * bzw. *.	Ja, Abgleich von path nach Präfix und von method und statusCode nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
Antwort	<pre>{ "location": { "type": "RESPONSE_BODY", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> ", "statusCode": "<i>status_code</i> " }, ... }</pre>	type	Die Standardwerte von path, method und statusCode sind /, * bzw. *.	Ja, Abgleich von path nach Präfix und von method und statusCode nach exakten Werten
Authorize	<pre>{ "location": { "type": "AUTHORIZER", "name": "<i>authorizer_name</i> " }, ... }</pre>	type	–	Nein
Model	<pre>{ "location": { "type": "MODEL", "name": "<i>model_name</i> " }, ... }</pre>	type	–	Nein

Dokumentationsversionen

Eine Dokumentationsversion ist eine Momentaufnahme der [DocumentationParts](#)-Sammlung einer API und ist mit einer Versionskennung gekennzeichnet. Die Veröffentlichung der Dokumentation einer API umfasst das Erstellen einer Dokumentationsversion, deren Verknüpfung mit einer API-Stufe und das Exportieren dieser stufenspezifischen Version der API-Dokumentation in eine externe OpenAPI-Datei. In API Gateway wird ein Dokumentations-Snapshot als [DocumentationVersion](#)-Ressource dargestellt.

Wenn Sie eine API aktualisieren, erstellen Sie neue Versionen der API. In API Gateway verwalten Sie alle Dokumentationsversionen mithilfe der [DocumentationVersions](#)-Sammlung.

Dokumentieren einer API mit der API Gateway-Konsole

In diesem Abschnitt wird beschrieben, wie Sie die Dokumentationsbausteine einer API mit der API Gateway-Konsole erstellen und verwalten.

Eine Voraussetzung für das Erstellen und Bearbeiten der Dokumentation einer API ist, dass Sie die API bereits erstellt haben. In diesem Abschnitt verwenden wir die [PetStore](#)-API als Beispiel. Um eine API mit der API Gateway-Konsole zu erstellen, befolgen Sie die Anweisungen in [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Themen

- [Dokumentieren der API-Entität](#)
- [Dokumentieren einer RESOURCE-Entität](#)
- [Dokumentieren einer METHOD-Entität](#)
- [Dokumentieren einer QUERY_PARAMETER-Entität](#)
- [Dokumentieren einer PATH_PARAMETER-Entität](#)
- [Dokumentieren einer REQUEST_HEADER-Entität](#)
- [Dokumentieren einer REQUEST_BODY-Entität](#)
- [Dokumentieren einer RESPONSE-Entität](#)
- [Dokumentieren einer RESPONSE_HEADER-Entität](#)
- [Dokumentieren einer RESPONSE_BODY-Entität](#)
- [Dokumentieren einer MODEL-Entität](#)
- [Dokumentieren einer AUTHORIZER-Entität](#)

Dokumentieren der **API**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für die API-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf **Documentation** (Dokumentation) und anschließend auf **Create Documentation Part** (Dokumentation erstellen).
2. Wählen Sie als **Documentation type** (Dokumentationstyp) die Option **API** aus.

Wenn kein Dokumentationsbaustein für die API erstellt wurde, wird der **properties**-Map-Editor des Dokumentationsbausteins angezeigt. Geben Sie die folgende **properties**-Zuweisung in den Texteditor ein.

```
{
  "info": {
    "description": "Your first API Gateway API.",
    "contact": {
      "name": "John Doe",
      "email": "john.doe@api.com"
    }
  }
}
```

Note

Sie müssen die **properties**-Zuordnung nicht als JSON-Zeichenfolge verschlüsseln. Die API Gateway-Konsole stringifiziert das JSON-Objekt für Sie.

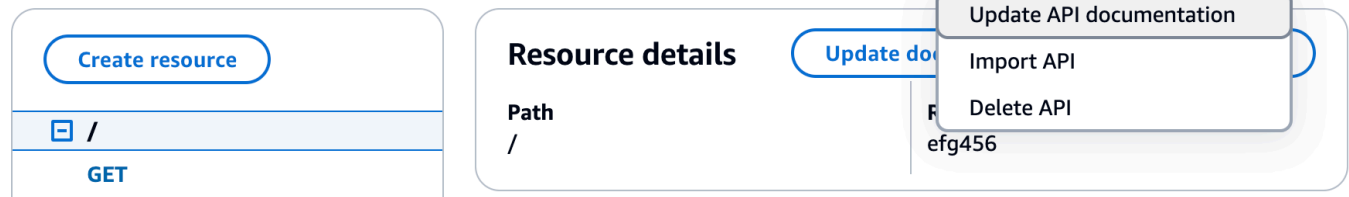
3. Wählen Sie **Create documentation part** (Dokumentation erstellen) aus.

Gehen Sie wie folgt vor, um eine neue Dokumentation für die API-Entität im Bereich **Resources** (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf **Resources** (Ressourcen).
2. Wählen Sie das Menü **API actions** (API-Aktionen) und dann **Update API documentation** (API-Dokumentation aktualisieren) aus.

[API Gateway](#) > [APIs](#) > [Resources - PetStore \(abcd1234\)](#)

Resources



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Wählen Sie den Namen Ihrer API aus und klicken Sie dann auf der API-Karte auf Edit (Bearbeiten).

Dokumentieren einer **RESOURCE**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine RESOURCE-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Resource (Ressource) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Geben Sie eine Beschreibung im Texteditor ein, zum Beispiel:

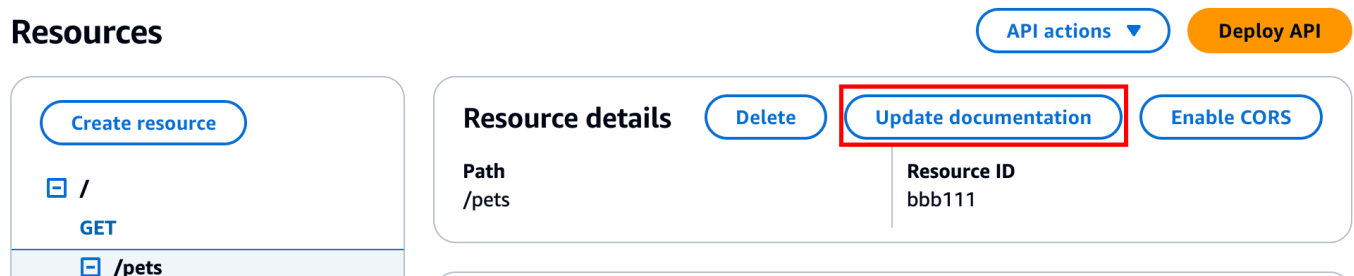
```
{
  "description": "The PetStore's root resource."
}
```

5. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für eine nicht aufgeführte Ressource erstellen.
6. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine RESOURCE-Entität im Bereich Resources (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
2. Wählen Sie die Ressource aus und klicken Sie dann auf Update documentation (Dokumentation aktualisieren).

Resources



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Wählen Sie die Ressource aus, die Ihre Dokumentation enthält, und klicken Sie dann auf Edit (Bearbeiten).

Dokumentieren einer **METHOD**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine METHOD-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Method (Methode) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie eine Beschreibung im Texteditor ein, zum Beispiel:

```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für eine nicht aufgeführte Methode erstellen.

7. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine METHOD-Entität im Bereich Resources (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
2. Wählen Sie die Methode aus und klicken Sie dann auf Update documentation (Dokumentation aktualisieren).

[API Gateway](#) > [APIs](#) > Resources - PetStore (abcd1234)

Resources

API actions ▼

Deploy API

Create resource

- /
 - GET
 - /pets
 - GET
 - OPTIONS
 - POST**
 - /petid
 - GET
 - OPTIONS

/pets - POST - Method execution

ARN
arn:aws:execute-api:us-west-2:111222333:abcd1234/*/POST/pets

Resource ID
aaa111

Update documentation **Delete**

Client → Method request → Integration request → HTTP integration

HTTP integration response ← Integration response ← Method response ← Client

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können die Methode oder die Ressource auswählen, die die Methode enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **QUERY_PARAMETER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine **QUERY_PARAMETER**-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf **Documentation** (Dokumentation) und anschließend auf **Create Documentation Part** (Dokumentation erstellen).
2. Wählen Sie als **Documentation type** (Dokumentationstyp) die Option **Query parameter** (Abfrageparameter) aus.
3. Geben Sie für **Path** (Pfad) einen Pfad ein.
4. Wählen Sie ein **HTTP-Verb** für **Method** (Methode) aus.
5. Geben Sie unter **Name** einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie **Create documentation part** (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführte Abfrageparameter erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich **Documentation** (Dokumentation) die Registerkarte **Resources and methods** (Ressourcen und Methoden) aus.
2. Sie können den Abfrageparameter oder die Ressource auswählen, die den Abfrageparameter enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie **Bearbeiten** aus.

Dokumentieren einer **PATH_PARAMETER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine **PATH_PARAMETER**-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf **Documentation** (Dokumentation) und anschließend auf **Create Documentation Part** (Dokumentation erstellen).
2. Wählen Sie als **Documentation type** (Dokumentationstyp) die Option **Path parameter** (Pfadparameter) aus.

3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Name einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführte Pfadparameter erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Pfadparameter oder die Ressource auswählen, die den Pfadparameter enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **REQUEST_HEADER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine REQUEST_HEADER-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Request header (Anforderungsheader) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Anforderungsheader ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Name einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Header erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Anforderungsheader oder die Ressource auswählen, die den Anforderungsheader enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **REQUEST_BODY**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine REQUEST_BODY-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Request body (Anforderungstext) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Anforderungstext ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie eine Beschreibung im Texteditor ein.
6. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Anforderungstext erstellen.
7. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Anforderungstext oder die Ressource auswählen, die den Anforderungstext enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **RESPONSE**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine RESPONSE-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen unter Documentation type (Dokumentationstyp) die Option Response (status code) (Antwort (Statuscode)) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für die Antwort ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antwort-Statuscode erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antwort-Statuscode oder die Ressource auswählen, die den Antwort-Statuscode enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **RESPONSE_HEADER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine RESPONSE_HEADER-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Response header (Antwortheader) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Antwortheader ein.

4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antwortheader erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antwortheader oder die Ressource auswählen, die den Antwortheader enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **RESPONSE_BODY**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine RESPONSE_BODY-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Response body (Antworttext) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Antworttext ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antworttext erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antworttext oder die Ressource auswählen, die den Antworttext enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **MODEL**-Entität

Das Dokumentieren einer MODEL-Entität beinhaltet das Erstellen und Verwalten von DocumentationPart-Instances für das Modell sowie der properties jedes Modells. Das Modell Error, das standardmäßig in jeder API enthalten ist, verfügt z. B. über die folgende Schemadefinition:

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

Es erfordert zwei DocumentationPart-Instances, eine für das Model und die andere für dessen message-Eigenschaft:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

und

```
{
```

```
"location": {
  "type": "MODEL",
  "name": "Error.message"
},
"properties": {
  "description": "An error message."
}
}
```

Wenn die API exportiert wird, überschreiben die Eigenschaften des `DocumentationPart` die Werte im ursprünglichen Schema.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine MODEL-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf **Documentation** (Dokumentation) und anschließend auf **Create Documentation Part** (Dokumentation erstellen).
2. Wählen Sie als **Documentation type** (Dokumentationstyp) die Option **Model** (Modell) aus.
3. Geben Sie für **Name** einen Modellnamen ein.
4. Geben Sie eine Beschreibung im Texteditor ein.
5. Wählen Sie **Create documentation part** (Dokumentation erstellen) aus. Sie können Dokumentation für nicht aufgeführte Modelle erstellen.
6. Wiederholen Sie ggf. diese Schritte, um einen Dokumentationsbaustein zu anderen Modellen hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine MODEL-Entität im Bereich **Models** (Modelle) hinzuzufügen:

1. Klicken Sie im Navigationsbereich auf **Models** (Modelle).
2. Wählen Sie das Modell aus und klicken Sie dann auf **Update documentation** (Dokumentation aktualisieren).

Models (6) Delete Edit Update documentation Create model

Use models to define the format for the body of different requests and responses used by your API.

	Name	Content type	Description
☐	Empty	application/json	
☐	NewPet	application/json	
☒	NewPetResponse	application/json	
☐	Pet	application/json	
☐	Pets	application/json	
☐	PetType	application/json	

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Models (Modelle) aus.
2. Verwenden Sie die Suchleiste oder wählen Sie das Modell aus und klicken Sie dann auf Edit (Bearbeiten).

Dokumentieren einer **AUTHORIZER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine AUTHORIZER-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Authorizer (Genehmiger) aus.
3. Geben Sie unter Name, einen Namen für den Genehmiger ein.
4. Geben Sie eine Beschreibung im Texteditor ein. Geben Sie einen Wert für das gültige location-Feld für den Genehmiger ein.
5. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für nicht aufgeführte Genehmiger erstellen.
6. Wiederholen Sie ggf. diese Schritte, um einen Dokumentationsbaustein zu anderen Genehmigern hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Authorizers (Genehmiger) aus.
2. Verwenden Sie die Suchleiste oder wählen Sie den Genehmiger aus und klicken Sie dann auf Edit (Bearbeiten).

Veröffentlichen der API-Dokumentation mithilfe der API Gateway-Konsole

Im folgenden Verfahren wird beschrieben, wie Sie eine Dokumentationsversion veröffentlichen.

So veröffentlichen Sie eine Dokumentationsversion mit der API Gateway-Konsole

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation).
2. Wählen Sie Publish documentation (Dokumentation veröffentlichen) aus.
3. Konfigurieren der Veröffentlichung:
 - a. Wählen Sie für Stufe eine Stufe aus.
 - b. Geben Sie für Version eine Versionskennung ein, z. B. 1.0.0.
 - c. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
4. Wählen Sie Publish.

Sie können nun die veröffentlichte Dokumentation herunterladen, indem Sie die Dokumentation in eine externe OpenAPI-Datei exportieren. Weitere Informationen hierzu finden Sie unter [the section called “Exportieren einer REST-API”](#).

Dokumentieren einer API mit der API Gateway-REST-API

In diesem Abschnitt wird beschrieben, wie Sie die Dokumentationsbausteine einer API mit der API Gateway-REST-API erstellen und verwalten.

Vor dem Erstellen und Bearbeiten der Dokumentation einer API erstellen Sie zunächst die API. In diesem Abschnitt verwenden wir die [PetStoreAPI](#) als Beispiel. Um eine API mit der API Gateway-Konsole zu erstellen, befolgen Sie die Anweisungen in [Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels](#).

Themen

- [Dokumentieren der API-Entität](#)
- [Dokumentieren einer RESOURCE-Entität](#)
- [Dokumentieren einer METHOD-Entität](#)
- [Dokumentieren einer QUERY_PARAMETER-Entität](#)
- [Dokumentieren einer PATH_PARAMETER-Entität](#)
- [Dokumentieren einer REQUEST_BODY-Entität](#)
- [Dokumentieren einer REQUEST_HEADER-Entität](#)
- [Dokumentieren einer RESPONSE-Entität](#)
- [Dokumentieren einer RESPONSE_HEADER-Entität](#)
- [Dokumentieren einer AUTHORIZER-Entität](#)
- [Dokumentieren einer MODEL-Entität](#)
- [Aktualisieren von Dokumentationsbausteinen](#)
- [Auflisten von Dokumentationsbausteinen](#)

Dokumentieren der **API**-Entität

Um Dokumentation für eine [API](#) hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource für die API-Entität hinzu:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "API"
  },
  "properties": "{\n\t\t\"info\" : {\n\t\t\t\t\"description\" : \"Your first API with Amazon
API Gateway.\n\t\t}\n\t}"
}
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  ...
  "id": "s2e5xf",
  "location": {
    "path": null,
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "API"
  },
  "properties": "{\n\t\t\"info\": {\n\t\t\t\t\"description\" : \"Your first API with Amazon
API Gateway.\n\t\t}\n}"
}
```

Wenn der Dokumentationsbaustein bereits hinzugefügt wurde, wird eine 409 Conflict-Antwort mit der Fehlermeldung `Documentation part already exists for the specified location: type 'API'.` zurückgegeben. In diesem Fall müssen Sie die Operation [documentationpart:update](#) aufrufen.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\t\"info\": {\n\t\t\t\t\"description\" : \"Your first API with Amazon API
Gateway.\n\t\t}\n}"
  } ]
}
```

Bei einer erfolgreichen Antwort wird der Statuscode 200 OK sowie eine Nutzlast mit der aktualisierten DocumentationPart-Instance in der Nutzlast zurückgegeben.

Dokumentieren einer **RESOURCE**-Entität

Um Dokumentation für die Root-Ressource einer API hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die auf die entsprechende [Ressourcenressource](#) ausgerichtet ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "RESOURCE",
  },
  "properties" : "{\n\t\"description\" : \"The PetStore root resource.\n\n}"
}
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    }
  },
}
```

```

    "id": "p76vqo",
    "location": {
      "path": "/",
      "method": null,
      "name": null,
      "statusCode": null,
      "type": "RESOURCE"
    },
    "properties": "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
  }

```

Wenn kein Ressourcenpfad angegeben ist, wird davon ausgegangen, dass die Ressource die Stammressource ist. Sie können "path": "/" zu properties hinzufügen, um diese Spezifikation explizit zu machen.

Um eine Dokumentation für eine untergeordnete Ressource einer API zu erstellen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die auf die entsprechende [Ressourcenressource](#) ausgerichtet ist:

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "RESOURCE",
    "path" : "/pets"
  },
  "properties": "{\n\t\"description\" : \"A child resource under the root of
PetStore.\"\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Beispiel:

```

{
  "_links": {
    "curies": {

```

```

    "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
    "name": "documentationpart",
    "templated": true
  },
  "self": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
  },
  "documentationpart:delete": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
  },
  "documentationpart:update": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
  }
},
"id": "qcht86",
"location": {
  "path": "/pets",
  "method": null,
  "name": null,
  "statusCode": null,
  "type": "RESOURCE"
},
"properties": "{\n\t\"description\" : \"A child resource under the root of PetStore.
\\\"\\n}\"
}

```

Um Dokumentation für eine untergeordnete Ressource hinzuzufügen, die durch einen Pfadparameter angegeben wird, fügen Sie eine [DocumentationPart](#) Ressource hinzu, die für die [Ressourcenressource](#) bestimmt ist:

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "RESOURCE",
    "path" : "/pets/{petId}"
  }
}

```

```

    },
    "properties": "{\n\t\"description\" : \"A child resource specified by the petId
path parameter.\"\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Beispiel:

```

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    }
  },
  "id": "k6fpwb",
  "location": {
    "path": "/pets/{petId}",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource specified by the petId path
parameter.\"\n}"
}

```

Note

Die [DocumentationPart](#) Instanz einer RESOURCE Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Dokumentieren einer METHOD-Entität

Um Dokumentation für eine Methode einer API hinzuzufügen, fügen Sie eine [DocumentationPart](#) Ressource hinzu, die auf die entsprechende [Methodenressource](#) ausgerichtet ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "METHOD",
    "path" : "/pets",
    "method" : "GET"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\n}"
}
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "documentationpart": {
    "location": {
      "type": "METHOD",
      "path": "/pets",
      "method": "GET"
    },
    "properties": "{\n\t\"summary\" : \"List all pets.\n}"
  }
}
```

```

    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",

```

```

    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\">\n}"
}
```

Wenn in der vorherigen Anforderung das Feld `location.method` nicht angegeben ist, wird davon ausgegangen, dass die ANY-Methode verwendet wird, dargestellt durch ein Platzhalterzeichen `*`.

Um den Dokumentationsinhalt einer METHOD-Entität zu aktualisieren, rufen Sie den Vorgang [documentationpart:update](#) auf, der eine neue properties-Zuordnung bereitstellt:

```

PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\">\n}"
  } ]
}
```

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit der aktualisierten DocumentationPart-Instance in der Nutzlast zurückgegeben. Zum Beispiel:

```

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
```

```

    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
}

```

Dokumentieren einer **QUERY_PARAMETER**-Entität

Um Dokumentation für einen Anforderungsabfrageparameter hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die auf den **QUERY_PARAMETER** Typ ausgerichtet ist und die gültigen Felder `path` und `name` enthält.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "QUERY_PARAMETER",
    "path" : "/pets",
    "method" : "GET",
    "name" : "page"
  },
  "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}

```


Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    }
  },
  "id": "h9ht5w",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "page",
    "statusCode": null,
    "type": "QUERY_PARAMETER"
  },
  "properties": "{\n\t\"description\" : \"Page number of results to return.\\\"\\n\""}
}
```

Die properties-Map des Dokumentationsbausteins einer QUERY_PARAMETER-Entität kann von einer der untergeordneten QUERY_PARAMETER-Entitäten übernommen werden. Wenn Sie z. B. eine treats-Ressource nach /pets/{petId} hinzufügen, die GET-Methode für /pets/{petId}/treats aktivieren und den page-Abfrageparameter bereitstellen, übernimmt dieser untergeordnete Abfrageparameter die DocumentationPart-Map des properties von dem gleichnamigen Abfrageparameter der GET /pets-Methode, es sei denn, Sie fügen explizit eine DocumentationPart-Ressource zum page-Abfrageparameter der GET /pets/{petId}/treats-Methode hinzu.

Dokumentieren einer **PATH_PARAMETER**-Entität

Um Dokumentation für einen Pfadparameter hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource für die PATH_PARAMETER Entität hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "PATH_PARAMETER",
    "path" : "/pets/{petId}",
    "method" : "*",
    "name" : "petId"
  },
  "properties": "{\n\t\"description\" : \"The id of the pet to retrieve.\"\n}"
}
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    }
  }
}
```

```

},
"id": "ckpgog",
"location": {
  "path": "/pets/{petId}",
  "method": "*",
  "name": "petId",
  "statusCode": null,
  "type": "PATH_PARAMETER"
},
"properties": "{\n  \"description\" : \"The id of the pet to retrieve\"\n}"
}

```

Dokumentieren einer **REQUEST_BODY**-Entität

Um Dokumentation für einen Anforderungstext hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource für den Anfragetext hinzu.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "REQUEST_BODY",
    "path" : "/pets",
    "method" : "POST"
  },
  "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",

```

```

    "templated": true
  },
  "self": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
  },
  "documentationpart:delete": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
  },
  "documentationpart:update": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
  }
},
"id": "kgmfr1",
"location": {
  "path": "/pets",
  "method": "POST",
  "name": null,
  "statusCode": null,
  "type": "REQUEST_BODY"
},
"properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}

```

Dokumentieren einer **REQUEST_HEADER**-Entität

Um Dokumentation für einen Anforderungsheader hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource für den Anforderungsheader hinzu.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "REQUEST_HEADER",
    "path" : "/pets",
    "method" : "GET",
    "name" : "x-my-token"
  },

```

```

    "properties": "{\n\t\"description\" : \"A custom token used to authorization the
method invocation.\n\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```

{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    }
  },
  "id": "h0m3uf",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "x-my-token",
    "statusCode": null,
    "type": "REQUEST_HEADER"
  },
  "properties": "{\n\t\"description\" : \"A custom token used to authorization the
method invocation.\n\n}"
}

```

Dokumentieren einer **RESPONSE**-Entität

Um eine Dokumentation für eine Antwort auf einen Statuscode hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die auf die entsprechende [MethodResponse](#)-Ressource ausgerichtet ist.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location": {
    "path": "/",
    "method": "*",
    "name": null,
    "statusCode": "200",
    "type": "RESPONSE"
  },
  "properties": "{\n  \"description\" : \"Successful operation.\\n\\n\"
}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Zum Beispiel:

```

{
  "_links": {
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
    }
  },
  "id": "lattew",
  "location": {
    "path": "/",
    "method": "*",
    "name": null,
    "statusCode": "200",
    "type": "RESPONSE"
  },
  "properties": "{\n  \"description\" : \"Successful operation.\\n\\n\"
}"
}

```

```
}
```

Dokumentieren einer **RESPONSE_HEADER**-Entität

Um Dokumentation für einen Antwort-Header hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource für den Antwort-Header hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{"location": {
  "path": "/",
  "method": "GET",
  "name": "Content-Type",
  "statusCode": "200",
  "type": "RESPONSE_HEADER"
},
"properties": "{\n  \"description\" : \"Media type of request\"\n}"
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    }
  }
}
```

```

    }
  },
  "id": "fev7j7",
  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
}

```

Die Dokumentation dieses Content-Type-Antwort-Headers ist die Standarddokumentation für die Content-Type-Header aller Antworten der API.

Dokumentieren einer **AUTHORIZER**-Entität

Um Dokumentation für einen API-Autorisierer hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die für den angegebenen Autorisierer bestimmt ist.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "AUTHORIZER",
    "name" : "myAuthorizer"
  },
  "properties": "{\n\t\"description\" : \"Authorizes invocations of configured
methods.\"\n}"
}

```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Beispiel:

```
{
```



```

"_links": {
  "curies": {
    "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
    "name": "documentationpart",
    "templated": true
  },
  "self": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
  },
  "documentationpart:delete": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
  },
  "documentationpart:update": {
    "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
  }
},
"id": "pw3qw3",
"location": {
  "path": null,
  "method": null,
  "name": "myAuthorizer",
  "statusCode": null,
  "type": "AUTHORIZER"
},
"properties": "{\n\t\"description\" : \"Authorizes invocations of configured methods.
\\\"\\n}\"
}

```

Note

Die [DocumentationPart](#) Instanz einer AUTHORIZER Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Dokumentieren einer **MODEL**-Entität

Das Dokumentieren einer MODEL-Entität beinhaltet das Erstellen und Verwalten von DocumentPart-Instances für das Modell sowie der properties jedes Modells. Das Modell Error, das standardmäßig in jeder API enthalten ist, verfügt z. B. über die folgende Schemadefinition:

```
{
```

```

"$schema" : "http://json-schema.org/draft-04/schema#",
"title" : "Error Schema",
"type" : "object",
"properties" : {
  "message" : { "type" : "string" }
}
}

```

Es erfordert zwei `DocumentationPart`-Instances, eine für das Modell und die andere für dessen `message`-Eigenschaft:

```

{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}

```

und

```

{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}

```

Wenn die API exportiert wird, überschreiben die Eigenschaften des `DocumentationPart` die Werte im ursprünglichen Schema.

Um Dokumentation für ein API-Modell hinzuzufügen, fügen Sie eine [DocumentationPart](#)-Ressource hinzu, die auf das angegebene Modell ausgerichtet ist.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
```

```
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "MODEL",
    "name" : "Pet"
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Bei Erfolg gibt die Operation eine 201 Created-Antwort zurück, die die neu erstellte DocumentationPart-Instance in der Nutzlast enthält. Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
    }
  },
  "id": "1kn4uq",
  "location": {
    "path": null,
    "method": null,
    "name": "Pet",
    "statusCode": null,
    "type": "MODEL"
  }
}
```

```

},
"properties": "{\n\t\"description\" : \"Data structure of a Pet object.\\n\\n}"
}

```

Wiederholen Sie denselben Schritt, um eine `DocumentationPart` Instanz für eine der Eigenschaften des Modells zu erstellen.

Note

Die [DocumentationPart](#) Instanz einer MODEL Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Aktualisieren von Dokumentationsbausteinen

Um die Dokumentationsteile jeder Art von API-Entitäten zu aktualisieren, reichen Sie eine PATCH-Anfrage für eine [DocumentationPart](#) Instanz einer bestimmten Teil-ID ein, um die vorhandene `properties` Map durch eine neue zu ersetzen.

```

PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}

```

Bei einer erfolgreichen Antwort wird der Statuscode 200 OK sowie eine Nutzlast mit der aktualisierten `DocumentationPart`-Instance in der Nutzlast zurückgegeben.

Sie können mehrere Dokumentationsbausteine in einer einzigen PATCH-Anforderung aktualisieren.

Auflisten von Dokumentationsbausteinen

Um die Dokumentationsteile jeder Art von API-Entitäten aufzulisten, reichen Sie eine GET-Anfrage für eine [DocumentationParts](#)Sammlung ein.

```
GET /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Bei einer erfolgreichen Antwort wird der Statuscode 200 OK sowie eine Nutzlast mit den verfügbaren DocumentationPart-Instances in der Nutzlast zurückgegeben.

Veröffentlichen der API-Dokumentation mithilfe der API Gateway-REST-API

Um die Dokumentation für eine API zu veröffentlichen, erstellen oder aktualisieren Sie einen Dokumentations-Snapshot bzw. rufen Sie einen Dokumentations-Snapshot ab und verknüpfen Sie diesen mit einer API-Stufe. Beim Erstellen eines Dokumentations-Snapshots können Sie ihn gleichzeitig mit einer API-Stufe verknüpfen.

Themen

- [Erstellen eines Dokumentations-Snapshots und dessen Verknüpfen mit einer API-Stufe](#)
- [Erstellen eines Dokumentations-Snapshots](#)
- [Aktualisieren eines Dokumentations-Snapshots](#)
- [Abrufen eines Dokumentations-Snapshots](#)
- [Verknüpfen eines Dokumentations-Snapshots mit einer API-Stufe](#)
- [Herunterladen eines mit einer Stufe verknüpften Dokumentations-Snapshots](#)

Erstellen eines Dokumentations-Snapshots und dessen Verknüpfen mit einer API-Stufe

Um einen Snapshot der API-Dokumentationsbausteine zu erstellen und ihn zugleich mit einer API-Stufe zu verknüpfen, übermitteln Sie die folgende POST-Anforderung:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
```

```

Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "documentationVersion" : "1.0.0",
  "stageName": "prod",
  "description" : "My API Documentation v1.0.0"
}

```

Bei Erfolg gibt die Operation eine 200 OK-Antwort zurück, die die neu erstellte `DocumentationVersion`-Instance als Nutzlast enthält.

Sie können auch einen Dokumentations-Snapshot erstellen, ohne ihn einer API-Stufe zuzuweisen, und dann [restapi:update](#) aufrufen, um den Snapshot mit einer bestimmten API-Stufe zu verknüpfen. Sie können auch einen vorhandene Dokumentations-Snapshot aktualisieren oder abfragen und dann dessen Verknüpfung ändern. Diese Schritte werden in den nächsten vier Abschnitten gezeigt.

Erstellen eines Dokumentations-Snapshots

Um einen Snapshot der Dokumentationsteile einer API zu erstellen, erstellen Sie eine neue [DocumentationVersion](#)-Ressource und fügen Sie sie der [DocumentationVersions](#)-Sammlung der API hinzu:

```

POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "documentationVersion" : "1.0.0",
  "description" : "My API Documentation v1.0.0"
}

```

Bei Erfolg gibt die Operation eine 200 OK-Antwort zurück, die die neu erstellte `DocumentationVersion`-Instance als Nutzlast enthält.

Aktualisieren eines Dokumentations-Snapshots

Sie können einen Dokumentations-Snapshot nur aktualisieren, indem Sie die `description` Eigenschaft der entsprechenden [DocumentationVersion](#) Ressource ändern. Im folgenden Beispiel wird gezeigt, wie Sie die Beschreibung des Dokumentations-Snapshots gemäß seiner Versionskennung, `version`, aktualisieren (z. B. 1.0.0).

```
PATCH /restapis/restapi_id/documentation/versions/version HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations": [{
    "op": "replace",
    "path": "/description",
    "value": "My API for testing purposes."
  }]
}
```

Bei Erfolg gibt die Operation eine 200 OK-Antwort zurück, die die aktualisierte `DocumentationVersion`-Instance als Nutzlast enthält.

Abrufen eines Dokumentations-Snapshots

Um eine Momentaufnahme der Dokumentation zu erhalten, reichen Sie eine GET Anfrage für die angegebene [DocumentationVersion](#) Ressource ein. Im folgenden Beispiel wird gezeigt, wie Sie einen Dokumentations-Snapshot für eine bestimmte Versionskennung (1.0.0) abrufen.

```
GET /restapis/<restapi_id>/documentation/versions/1.0.0 HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Verknüpfen eines Dokumentations-Snapshots mit einer API-Stufe

Um die API-Dokumentation zu veröffentlichen, verknüpfen Sie einen Dokumentations-Snapshot mit einer API-Stufe. Sie müssen bereits eine API-Stufe erstellt haben, bevor Sie die Dokumentationsversion damit verknüpfen können.

Um einen Dokumentations-Snapshot mit einer API-Stufe über die [API Gateway-REST-API](#) zu verknüpfen, rufen Sie den Vorgang [stage:update](#) auf, um die gewünschte Dokumentationsversion für die Eigenschaft `stage.documentationVersion` festzulegen:

```
PATCH /restapis/RESTAPI_ID/stages/STAGE_NAME
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations": [{
    "op": "replace",
    "path": "/documentationVersion",
    "value": "VERSION_IDENTIFIER"
  }]
}
```

Herunterladen eines mit einer Stufe verknüpften Dokumentations-Snapshots

Nachdem eine Version der Dokumentationsteile einer Phase zugeordnet wurde, können Sie die Dokumentationsteile zusammen mit den API-Entitätsdefinitionen in eine externe Datei exportieren, indem Sie die API Gateway-Konsole, die API Gateway Gateway-REST-API SDKs, eine davon oder die AWS CLI für API Gateway verwenden. Der Vorgang ist mit dem für das Exportieren der API identisch. Das exportierte Dateiformat kann JSON oder YAML sein.

Mit der API Gateway-REST-API können Sie auch explizit den Abfrageparameter `extension=documentation,integrations,authorizers` so konfigurieren, dass die API-Dokumentationsbausteine, API-Integrationen und Genehmiger beim API-Export berücksichtigt werden. Standardmäßig sind die Dokumentationsbausteine enthalten, die Integrationen und Genehmiger aber vom Export einer API ausgeschlossen. Die Standardausgabe von einem API-Export eignet sich für die Verteilung der Dokumentation.

Um die API-Dokumentation in eine externe JSON-OpenAPI-Datei mit der API Gateway-REST-API zu exportieren, übermitteln Sie die folgende GET-Anforderung:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Hier enthält das Objekt `x-amazon-apigateway-documentation` die Dokumentationsbausteine und die API-Entitätsdefinitionen beinhalten die von OpenAPI unterstützten Dokumentationseigenschaften. Die Ausgabe enthält keine Details zur Integration oder zu Lambda-Genehmigern (ehemals als benutzerdefinierte Genehmiger bezeichnet). Um beide Details einzuschließen, konfigurieren Sie `extensions=integrations,authorizers,documentation`. Um Details zu Integrationen, aber nicht zu Genehmigern einzuschließen, konfigurieren Sie `extensions=integrations,documentation`.

Sie müssen den `Accept:application/json`-Header in der Anforderung so konfigurieren, dass die Ausgabe in einer JSON-Datei erfolgt. Um eine YAML-Ausgabe zu erzeugen, ändern den Header der Anforderung in `Accept:application/yaml`.

Als Beispiel wird eine API verwendet, die eine einfache GET-Methode für die Stammressource (/) bereitstellt. Diese API verfügt über vier API-Entitäten, die in einer OpenAPI-Definitionsdatei definiert sind, eine für jeden der Typen API, MODEL, METHOD und RESPONSE. Jeder der API-, METHOD- und RESPONSE-Entitäten wurde ein Dokumentationsbaustein hinzugefügt. Durch Aufrufen des vorherigen Befehls zum Exportieren der Dokumentation erhalten wir die folgende Ausgabe, wobei die Dokumentationsbausteine im Objekt `x-amazon-apigateway-documentation` als Erweiterung einer OpenAPI-Standarddatei aufgeführt werden.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "description": "API info description",
    "version": "2016-11-22T22:39:14Z",
```

```

        "title": "doc",
        "x-bar": "API info x-bar"
    },
    "paths": {
        "/": {
            "get": {
                "description": "Method description.",
                "responses": {
                    "200": {
                        "description": "200 response",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "$ref": "#/components/schemas/Empty"
                                }
                            }
                        }
                    }
                }
            },
            "x-example": "x- Method example"
        },
        "x-bar": "resource x-bar"
    }
},
"x-amazon-apigateway-documentation": {
    "version": "1.0.0",
    "createdDate": "2016-11-22T22:41:40Z",
    "documentationParts": [
        {
            "location": {
                "type": "API"
            },
            "properties": {
                "description": "API description",
                "foo": "API foo",
                "x-bar": "API x-bar",
                "info": {
                    "description": "API info description",
                    "version": "API info version",
                    "foo": "API info foo",
                    "x-bar": "API info x-bar"
                }
            }
        }
    ]
},

```

```

    {
      "location": {
        "type": "METHOD",
        "method": "GET"
      },
      "properties": {
        "description": "Method description.",
        "x-example": "x- Method example",
        "foo": "Method foo",
        "info": {
          "version": "method info version",
          "description": "method info description",
          "foo": "method info foo"
        }
      }
    },
    {
      "location": {
        "type": "RESOURCE"
      },
      "properties": {
        "description": "resource description",
        "foo": "resource foo",
        "x-bar": "resource x-bar",
        "info": {
          "description": "resource info description",
          "version": "resource info version",
          "foo": "resource info foo",
          "x-bar": "resource info x-bar"
        }
      }
    }
  ],
  "x-bar": "API x-bar",
  "servers": [
    {
      "url": "https://rznapp68yi.execute-api.ap-southeast-1.amazonaws.com/{basePath}",
      "variables": {
        "basePath": {
          "default": "/test"
        }
      }
    }
  ]
}

```

```

    }
  ],
  "components": {
    "schemas": {
      "Empty": {
        "type": "object",
        "title": "Empty Schema"
      }
    }
  }
}

```

OpenAPI 2.0

```

{
  "swagger" : "2.0",
  "info" : {
    "description" : "API info description",
    "version" : "2016-11-22T22:39:14Z",
    "title" : "doc",
    "x-bar" : "API info x-bar"
  },
  "host" : "rznaap68yi.execute-api.ap-southeast-1.amazonaws.com",
  "basePath" : "/test",
  "schemes" : [ "https" ],
  "paths" : {
    "/" : {
      "get" : {
        "description" : "Method description.",
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            }
          }
        }
      },
      "x-example" : "x- Method example"
    },
    "x-bar" : "resource x-bar"
  }
},

```

```
"definitions" : {
  "Empty" : {
    "type" : "object",
    "title" : "Empty Schema"
  }
},
"x-amazon-apigateway-documentation" : {
  "version" : "1.0.0",
  "createdDate" : "2016-11-22T22:41:40Z",
  "documentationParts" : [ {
    "location" : {
      "type" : "API"
    },
    "properties" : {
      "description" : "API description",
      "foo" : "API foo",
      "x-bar" : "API x-bar",
      "info" : {
        "description" : "API info description",
        "version" : "API info version",
        "foo" : "API info foo",
        "x-bar" : "API info x-bar"
      }
    }
  }, {
    "location" : {
      "type" : "METHOD",
      "method" : "GET"
    },
    "properties" : {
      "description" : "Method description.",
      "x-example" : "x- Method example",
      "foo" : "Method foo",
      "info" : {
        "version" : "method info version",
        "description" : "method info description",
        "foo" : "method info foo"
      }
    }
  }, {
    "location" : {
      "type" : "RESOURCE"
    },
    "properties" : {
```

```

    "description" : "resource description",
    "foo" : "resource foo",
    "x-bar" : "resource x-bar",
    "info" : {
      "description" : "resource info description",
      "version" : "resource info version",
      "foo" : "resource info foo",
      "x-bar" : "resource info x-bar"
    }
  }
} ]
},
"x-bar" : "API x-bar"
}

```

Bei einem OpenAPI-kompatiblen Attribut, das in der `properties`-Map eines Dokumentationsbausteins definiert ist, fügt API Gateway das Attribut in die zugehörige API-Entitätsdefinition ein. Das Attribut `x-something` ist eine OpenAPI-Standarderweiterung. Diese Erweiterung wird in die API-Entitätsdefinition propagiert. Schauen Sie sich z. B. das Attribut `x-example` für die GET-Methode an. Ein Attribut wie `foo` ist nicht Teil der OpenAPI-Spezifikation und wird nicht in die dazugehörigen API-Entitätsdefinitionen eingefügt.

Wenn ein Dokumentations-Rendering-Tool (z. B. [OpenAPI UI](#)) die API-Entitätsdefinitionen analysiert, um Dokumentationsattribute zu extrahieren, sind alle nicht mit OpenAPI kompatiblen `properties`-Attribute einer `DocumentationPart`-Instance nicht für das Tool verfügbar. Wenn jedoch ein Dokumentations-Rendering-Tool das Objekt `x-amazon-apigateway-documentation` analysiert, um Inhalte zu erhalten, oder wenn das Tool [restapi:documentation-parts](#) und [documentationpart:by-id](#) aufruft, um Dokumentationsbausteine von API Gateway abzurufen, sind alle Dokumentationsattribute für das Tool zur Anzeige verfügbar.

Um die Dokumentation mit API-Entitätsdefinitionen, die Integrationsdetails enthalten, in eine JSON-OpenAPI-Datei zu exportieren, übermitteln Sie folgende GET-Anforderung:

```

GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,documentation HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ

```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Um die Dokumentation mit API-Entitätsdefinitionen, die Details zu Integrationen und Genehmigern enthalten, in eine YAML-OpenAPI-Datei zu exportieren, übermitteln Sie die folgende GET-Anforderung:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,authorizers,documentation HTTP/1.1
Accept: application/yaml
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Um die API Gateway-Konsole zum Exportieren und Herunterladen der veröffentlichten Dokumentation einer API zu verwenden, befolgen Sie die Anweisungen unter [REST-API über die API-Gateway-Konsole exportieren](#).

Importieren einer API-Dokumentation

Wie beim Importieren von API-Entitätsdefinitionen können Sie Dokumentationsbausteine von einer externen OpenAPI-Datei in eine API in API Gateway importieren. Sie geben die to-be-imported Dokumentationsteile innerhalb der [x-amazon-apigateway-documentation Objekt](#) Erweiterung in einer gültigen OpenAPI-Definitionsdatei an. Durch das Importieren der Dokumentation werden die vorhandenen API-Entitätsdefinitionen nicht geändert.

Sie haben die Möglichkeit, die neu angegebenen Dokumentationsbausteine in vorhandene Dokumentationsbausteine in API Gateway zusammenzuführen oder die vorhandenen Dokumentationsbausteine zu überschreiben. Im MERGE-Modus wird ein neuer Dokumentationsbaustein, der in der OpenAPI-Datei definiert ist, der DocumentationParts-Sammlung der API hinzugefügt. Wenn bereits ein importierter DocumentationPart vorhanden ist, ersetzt ein importiertes Attribut jeweils einen vorhandenen Baustein, sofern einer der beiden unterschiedlich ist. Andere vorhandene Dokumentationsattribute bleiben davon unberührt. Im Modus OVERWRITE wird die gesamte DocumentationParts-Sammlung entsprechend der importierten OpenAPI-Definitionsdatei ersetzt.

Importieren der Dokumentationsbausteine mit der API Gateway-REST-API

Um eine API-Dokumentation mit der API Gateway-REST-API zu importieren, rufen Sie den Vorgang [documentationpart:import](#) auf. Das folgende Beispiel zeigt, wie vorhandene Dokumentationsbausteine einer API mit einer einzigen GET / -Methode überschrieben werden. Bei Erfolg wird eine 200 OK-Antwort zurückgegeben.

OpenAPI 3.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "openapi": "3.0.0",
  "info": {
    "description": "description",
    "version": "1",
    "title": "doc"
  },
  "paths": {
    "/": {
      "get": {
        "description": "Method description.",
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        }
      }
    }
  }
},
```



```
"x-amazon-apigateway-documentation": {
  "version": "1.0.3",
  "documentationParts": [
    {
      "location": {
        "type": "API"
      },
      "properties": {
        "description": "API description",
        "info": {
          "description": "API info description 4",
          "version": "API info version 3"
        }
      }
    },
    {
      "location": {
        "type": "METHOD",
        "method": "GET"
      },
      "properties": {
        "description": "Method description."
      }
    },
    {
      "location": {
        "type": "MODEL",
        "name": "Empty"
      },
      "properties": {
        "title": "Empty Schema"
      }
    },
    {
      "location": {
        "type": "RESPONSE",
        "method": "GET",
        "statusCode": "200"
      },
      "properties": {
        "description": "200 response"
      }
    }
  ]
}
```

```

    },
    "servers": [
      {
        "url": "/"
      }
    ],
    "components": {
      "schemas": {
        "Empty": {
          "type": "object",
          "title": "Empty Schema"
        }
      }
    }
  }
}

```

OpenAPI 2.0

```

PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

```

```

{
  "swagger": "2.0",
  "info": {
    "description": "description",
    "version": "1",
    "title": "doc"
  },
  "host": "",
  "basePath": "/",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "description": "Method description.",
        "produces": [

```

```
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          }
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
      {
        "location": {
          "type": "API"
        },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
        "location": {
          "type": "METHOD",
          "method": "GET"
        },
        "properties": {
          "description": "Method description."
        }
      }
    ]
  },
  {
```

```

    "location": {
      "type": "MODEL",
      "name": "Empty"
    },
    "properties": {
      "title": "Empty Schema"
    }
  },
  {
    "location": {
      "type": "RESPONSE",
      "method": "GET",
      "statusCode": "200"
    },
    "properties": {
      "description": "200 response"
    }
  }
]
}
}

```

Bei Erfolg gibt diese Anforderung eine "200 OK"-Antwort zurück, die die importierte `DocumentationPartId` in der Nutzlast enthält.

```

{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}

```

Sie können auch [restapi:import](#) oder [restapi:put](#) aufrufen. Die Dokumentationsbausteine im Objekt `x-amazon-apigateway-documentation` werden dann als Teil der OpenAPI-Eingabedatei der API-Definition bereitgestellt. Um die Dokumentationsbausteine vom API-Import auszuschließen, konfigurieren Sie `ignore=documentation` in den Abfrageparametern der Anforderung.

Importieren der Dokumentationsbausteine mit der API Gateway-Konsole

In den folgenden Anweisungen wird beschrieben, wie Sie Dokumentationsbausteine importieren.

So verwenden Sie die Konsole, um Dokumentationsbausteine einer API aus einer externen Datei zu importieren

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation).
2. Wählen Sie Importieren aus.
3. Wenn Sie bereits über eine Dokumentation verfügen, wählen Sie Overwrite (Überschreiben) oder Merge (Zusammenführen) aus, um Ihre neue Dokumentation entweder zu überschreiben oder zusammenzuführen.
4. Wählen Sie Choose file (Datei auswählen) aus, um eine Datei von einem Laufwerk zu laden, oder geben Sie die Inhalte einer Datei in die Dateiansicht ein. Ein Beispiel finden Sie in der Nutzlast der Beispielanforderung unter [Importieren der Dokumentationsbausteine mit der API Gateway-REST-API](#).
5. Wählen Sie aus, wie mit Warnungen beim Import umgegangen werden soll. Wählen Sie entweder Fail on warnings (Bei Warnungen fehlschlagen) oder Ignore warnings (Warnungen ignorieren). Weitere Informationen finden Sie unter [the section called “Fehler und Warnungen beim Import Ihrer API in API Gateway”](#).
6. Wählen Sie Importieren aus.

Zugriff auf API-Dokumentation in API Gateway steuern

Wenn Sie über ein eigenes Dokumentationsteam verfügen, das Ihre API-Dokumentation verfasst und bearbeitet, können Sie separate Zugriffsberechtigungen für Ihre Entwickler (für die API-Entwicklung) und für Ihre Autoren und Editoren (für die Entwicklung von Inhalten) konfigurieren. Dies ist besonders dann sinnvoll, wenn ein Drittanbieter an der Erstellung der Dokumentation beteiligt ist.

Um Ihrem Dokumentationsteam Zugriff auf die Erstellung, Aktualisierung und Veröffentlichung Ihrer API-Dokumentation zu gewähren, können Sie dem Dokumentationsteam eine IAM-Rolle mit der folgenden IAM-Richtlinie zuweisen, wobei die AWS Konto-ID Ihres Dokumentationsteams angegeben *account_id* ist.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Sid": "StmtDocPartsAddEditViewDelete",
  "Effect": "Allow",
  "Action": [
    "apigateway:GET",
    "apigateway:PUT",
    "apigateway:POST",
    "apigateway:PATCH",
    "apigateway:DELETE"
  ],
  "Resource": [
    "arn:aws:apigateway:us-east-1:111111111111:/restapis/*/documentation/*"
  ]
}
```

Weitere Informationen zum Konfigurieren von Berechtigungen für den Zugriff auf API Gateway-Ressourcen finden Sie unter [the section called “Funktionsweise von Amazon API Gateway mit IAM”](#).

Generieren Sie SDKs für REST APIs in API Gateway

Um Ihre REST-API auf plattform- oder sprachenspezifische Weise aufzurufen, müssen Sie das plattform- oder sprachenspezifische SDK der API generieren. Sie generieren Ihr SDK nachdem Sie Ihre API in einer Stufe erstellt, getestet und bereitgestellt haben. Derzeit unterstützt API Gateway die Generierung eines SDK für eine API in Java, Java für Android JavaScript, Objective-C oder Swift für iOS und Ruby.

In diesem Abschnitt wird erläutert, wie ein SDK einer API Gateway-API generiert wird. Es zeigt auch, wie das generierte SDK in einer Java-App, einer Java-App für Android, Objective-C- und Swift für iOS-Apps und einer JavaScript App verwendet wird.

Zur Vereinfachung der Erläuterung verwenden wir die API Gateway-[API](#), die diese [einfache Lambda-Rechnerfunktion](#) bereitstellt.

Bevor Sie fortfahren, erstellen oder importieren Sie die API und stellen sie mindestens einmal in API Gateway bereit. Detaillierte Anweisungen finden Sie unter [REST-APIs in API Gateway bereitstellen](#).

Themen

- [Einfache Lambda-Rechnerfunktion](#)

- [Einfache Rechner-API in API Gateway](#)
- [OpenAPI-Definition für einfache Rechner-API](#)
- [Generieren des Java-SDK einer API in API Gateway](#)
- [Generieren des Android SDK einer API in API Gateway](#)
- [Generieren des iOS SDK einer API in API Gateway](#)
- [Generieren Sie das JavaScript SDK einer REST-API in API Gateway](#)
- [Generieren des Ruby SDK einer API in API Gateway](#)
- [Generieren Sie SDKs für eine API mithilfe von AWS CLI Befehlen in API Gateway](#)

Einfache Lambda-Rechnerfunktion

Zur Veranschaulichung verwenden wir eine Node.js-Lambda-Funktion, die die binären Vorgänge Addition, Subtraktion, Multiplikation und Division durchführt.

Themen

- [Eingabeformat der einfachen Lambda-Rechnerfunktion](#)
- [Ausgabeformat der einfachen Lambda-Rechnerfunktion](#)
- [Implementieren der einfachen Lambda-Rechnerfunktion](#)

Eingabeformat der einfachen Lambda-Rechnerfunktion

Diese Funktion nimmt eine Eingabe im folgenden Format entgegen:

```
{ "a": "Number", "b": "Number", "op": "string" }
```

wobei op für jeden der Werte (+, -, *, /, add, sub, mul, div) stehen kann.

Ausgabeformat der einfachen Lambda-Rechnerfunktion

Wenn ein Vorgang erfolgreich ist, wird das Ergebnis im folgenden Format zurückgegeben:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number" }
```

wobei c das Ergebnis der Berechnung enthält.

Implementieren der einfachen Lambda-Rechnerfunktion

Die Implementierung der Lambda-Funktion wird wie folgt durchgeführt:

```
export const handler = async function (event, context) {
  console.log("Received event:", JSON.stringify(event));

  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
    return "400 Invalid Input";
  }

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
    case "add":
      res.c = res.a + res.b;
      break;
    case "-":
    case "sub":
      res.c = res.a - res.b;
      break;
    case "*":
    case "mul":
      res.c = res.a * res.b;
      break;
    case "/":
    case "div":
      if (res.b == 0) {
        return "400 Divide by Zero";
      } else {
        res.c = res.a / res.b;
      }
      break;
    default:
```

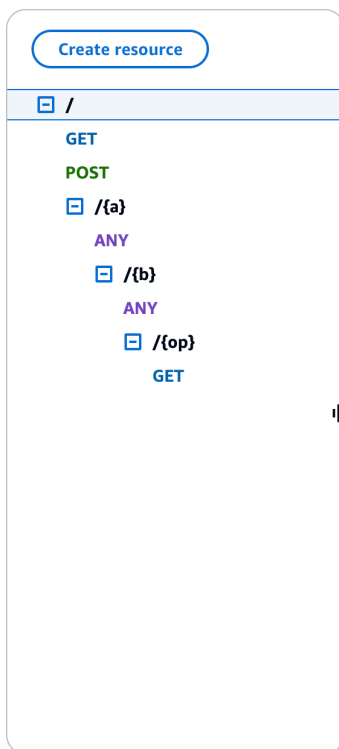


```
    return "400 Invalid Operator";  
  }  
  
  return res;  
};
```

Einfache Rechner-API in API Gateway

Unsere einfache Rechner-API stellt drei Methoden (GET, POST, GET) zum Aufrufen der berei [the section called “Einfache Lambda-Rechnerfunktion”](#). Eine grafische Darstellung dieser API sieht wie folgt aus:

Resources



Diese drei Methoden zeigen verschiedene Methoden zur Bereitstellung der Eingabe für die Backend-Lambda-Funktion, um denselben Vorgang auszuführen:

- Die GET `/?a=...&b=...&op=...`-Methode verwendet die Abfrageparameter, um den Eingang anzugeben.
- Die POST `/-`-Methode verwendet die JSON-Nutzlast `{"a": "Number", "b": "Number", "op": "string"}`, um den Eingang anzugeben.
- Die GET `/{a}/{b}/{op}`-Methode verwendet die Pfadparameter, um den Eingang anzugeben.

Ist keine Definition vorhanden, generiert API Gateway den entsprechenden SDK-Methodennamen durch eine Kombination der HTTP-Methode und den Pfadbestandteilen. Der Root-Pfadbestandteil (/) wird als `ApiRoot` bezeichnet. Beispiel: Der Standardname der Java-SDK-Methoden für die API-Methode von `GET /?a=...&b=...&op=...` lautet `getABOp`, der Standardname der SDK-Methode für `POST /` lautet `postApiRoot` und der Standardname der SDK-Methode für `GET /{a}/{b}/{op}` lautet `getABOp`. Der Einzelne SDKs kann die Konvention anpassen. Weitere Informationen finden Sie in der Dokumentation in der generierten SDK-Quelle für SDK-spezifische Methodennamen.

Sie können, und sollten, die Standardnamen der SDK-Methode überschreiben, indem Sie die Eigenschaft `operationName` für jede API-Methode angeben. Dies ist bei [Erstellung der API-Methode](#) oder der [Aktualisierung der API-Methode](#) mit der API Gateway-REST-API möglich. In der API-Swagger-Definition können Sie die `operationId` festlegen, um dasselbe Ergebnis zu erzielen.

Bevor wir zeigen, wie diese Methoden mit einem SDK aufgerufen werden, das von API Gateway für diese API generiert wurde, fassen wir die Einrichtung kurz zusammen. Detaillierte Anweisungen finden Sie unter [Entwickeln einer REST-API in API Gateway](#). Wenn API Gateway für Sie neu ist, lesen Sie zuerst [Wählen Sie ein AWS Lambda Integrations-Tutorial](#).

Erstellen von Modellen für Eingaben und Ausgaben

Wir müssen ein Input-Modell für die API erstellen, um stark typisierte Eingaben im SDK anzugeben. Für die Beschreibung des Antworttextdatentyps erstellen wir ein Output-Modell und ein Result-Modell.

Erstellen von Modellen für Eingabe, Ausgabe und Ergebnis

1. Klicken Sie im Navigationsbereich auf **Models (Modelle)**.
2. Wählen Sie **Modell erstellen** aus.
3. Geben Sie unter **Name** **input** ein.
4. Geben Sie für **Content type (Inhaltstyp)** **application/json** ein.

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **\$default** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.

5. Geben Sie für **Modellschema** Folgendes ein:

```
{
  "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
```

```
"type": "object",
"properties": {
  "a": { "type": "number" },
  "b": { "type": "number" },
  "op": { "type": "string" }
},
"title": "Input"
}
```

6. Wählen Sie Modell erstellen aus.
7. Wiederholen Sie die folgenden Schritte, um jeweils ein Output- und ein Result-Modell zu erstellen.

Geben Sie für das Output-Modell Folgendes für Modellschema ein:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "c": { "type": "number" }
  },
  "title": "Output"
}
```

Geben Sie für das Result-Modell Folgendes für Modellschema ein: Ersetzen Sie die API-ID abc123 durch Ihre API-ID.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "input": {
      "$ref": "https://apigateway.amazonaws.com/restapis/abc123/models/Input"
    },
    "output": {
      "$ref": "https://apigateway.amazonaws.com/restapis/abc123/models/Output"
    }
  },
  "title": "Result"
}
```

Einrichten von GET-/Abfrageparameter-Methode

Bei der GET `/?a=...&b=...&op=...`-Methode werden die Abfrageparameter in Method Request angegeben:

Einrichten der GET/URL-Abfragezeichenfolgenparameter

1. Klicken Sie im Abschnitt Methodenanforderung für die GET-Methode in der Stamm-Ressource (/) auf Bearbeiten.
2. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:
 - a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
 - b. Geben Sie unter Name **a** ein.
 - c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
 - d. Caching bleibt ausgeschaltet.

Wiederholen Sie die obigen Schritte und erstellen Sie eine Abfragezeichenfolge mit dem Namen **b** sowie eine Abfragezeichenfolge mit dem Namen **op**.

3. Wählen Sie Speichern.

Einrichten eines Datenmodells für die Nutzlast als Eingabe für das Backend

Bei der POST `/`-Methode erstellen wir das Input-Modell und fügen Sie es der Methodenabfrage hinzu, um die Form der Eingabedaten zu definieren.

Einrichten des Datenmodells für die Nutzlast als Eingabe für das Backend

1. Klicken Sie im Abschnitt Methodenanforderung für die POST-Methode in der Stamm-Ressource (/) auf Bearbeiten.
2. Klicken Sie auf Anforderungstext.
3. Wählen Sie Add model aus.
4. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
5. Wählen Sie für Modell die Option Eingabe aus.
6. Wählen Sie Speichern.

Bei diesem Modell können Ihre API-Kunden das SDK aufrufen, um die Eingabe eines Input-Objekts zu instanziiieren. Ohne dieses Modell müssten die Kunden ein Dictionary-Objekt erstellen, um die JSON-Eingabe in der Lambda-Funktion darzustellen.

Einrichten eines Datenmodells für die Ergebnisausgabe vom Backend

Für alle drei Methoden erstellen wir das `Result`-Modell und fügen es zur Method `Response` der Methode hinzu, um die Form der Ausgabe zu definieren, die von der Lambda-Funktion zurückgegeben wird.

Einrichten des Datenmodells für die Ergebnisausgabe aus dem Backend

1. Wählen Sie die Ressource `/a/b/op` und dann die GET-Methode aus.
2. Klicken Sie auf der Registerkarte Methodenantwort unter Antwort 200 auf Bearbeiten.
3. Klicken Sie unter Anforderungstext auf Modell hinzufügen.
4. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
5. Wählen Sie für Modell die Option Ergebnis aus.
6. Wählen Sie Speichern.

Bei diesem Modell können Ihre API-Kunden eine erfolgreiche Ausgabe parsen, indem sie die Eigenschaften für ein `Result`-Objekt lesen. Ohne dieses Modell müssten die Kunden ein Dictionary-Objekt erstellen, um die JSON-Ausgabe darzustellen.

OpenAPI-Definition für einfache Rechner-API

Nachfolgend finden Sie die OpenAPI-Definition für eine einfache Rechner-API. Diese können Sie in Ihr Konto importieren. Nach dem Import müssen jedoch die ressourcenbasierten Berechtigungen für die [Lambda-Funktion](#) zurückgesetzt werden. Dazu wählen Sie die für Ihr Konto erstellte Lambda-Funktion in der API Gateway-Konsole unter Integration Request (Integrationsanforderung) erneut aus. Dann setzt die API Gateway-Konsole die erforderlichen Berechtigungen zurück. Alternativ können Sie die AWS Command Line Interface für den Lambda-Befehl [add-permission](#) verwenden.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-29T20:27:30Z",
    "title": "SimpleCalc"
```

```
},
"host": "t6dve4zn25.execute-api.us-west-2.amazonaws.com",
"basePath": "/demo",
"schemes": [
  "https"
],
"paths": {
  "/": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "op",
          "in": "query",
          "required": false,
          "type": "string"
        },
        {
          "name": "a",
          "in": "query",
          "required": false,
          "type": "string"
        },
        {
          "name": "b",
          "in": "query",
          "required": false,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Result"
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
```

```

    "requestTemplates": {
      "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
  \"$input.params('op')\"\n}"
    },
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
    "passthroughBehavior": "when_no_templates",
    "httpMethod": "POST",
    "responses": {
      "default": {
        "statusCode": "200",
        "responseTemplates": {
          "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
  \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
        }
      }
    },
    "type": "aws"
  }
},
"post": {
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "in": "body",
      "name": "Input",
      "required": true,
      "schema": {
        "$ref": "#/definitions/Input"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Result"
      }
    }
  }
}

```

```

    }
  },
  "x-amazon-apigateway-integration": {
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST",
    "responses": {
      "default": {
        "statusCode": "200",
        "responseTemplates": {
          "application/json": "#set($inputRoot = $input.path('$'))\n{\n
\n\"input\" : {\n  \"a\" : $inputRoot.a,\n  \"b\" : $inputRoot.b,\n  \"op\" :
\n\"$inputRoot.op\"\n },\n  \"output\" : {\n  \"c\" : $inputRoot.c\n }\n}"
        }
      }
    },
    "type": "aws"
  }
},
"/{a}": {
  "x-amazon-apigateway-any-method": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "a",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "404": {
        "description": "404 response"
      }
    },
    "x-amazon-apigateway-integration": {

```



```

    "requestTemplates": {
      "application/json": "{\"statusCode\": 200}"
    },
    "passthroughBehavior": "when_no_match",
    "responses": {
      "default": {
        "statusCode": "404",
        "responseTemplates": {
          "application/json": "{ \"Message\" : \"Can't $context.httpMethod  
$context.resourcePath\" }"
        }
      }
    },
    "type": "mock"
  }
},
"/{a}/{b}": {
  "x-amazon-apigateway-any-method": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "a",
        "in": "path",
        "required": true,
        "type": "string"
      },
      {
        "name": "b",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "404": {
        "description": "404 response"
      }
    }
  },

```

```
"x-amazon-apigateway-integration": {
  "requestTemplates": {
    "application/json": "{\"statusCode\": 200}"
  },
  "passthroughBehavior": "when_no_match",
  "responses": {
    "default": {
      "statusCode": "404",
      "responseTemplates": {
        "application/json": "{ \"Message\" : \"Can't $context.httpMethod  
$context.resourcePath\" }"
      }
    }
  },
  "type": "mock"
}
},
"/{a}/{b}/{op}": {
  "get": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "a",
        "in": "path",
        "required": true,
        "type": "string"
      },
      {
        "name": "b",
        "in": "path",
        "required": true,
        "type": "string"
      },
      {
        "name": "op",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ]
  }
}
```

```

    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Result"
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "requestTemplates": {
      "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
  \"$input.params('op')\"\n}"
    },
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
    "passthroughBehavior": "when_no_templates",
    "httpMethod": "POST",
    "responses": {
      "default": {
        "statusCode": "200",
        "responseTemplates": {
          "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
  \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
        }
      }
    },
    "type": "aws"
  }
}
},
"definitions": {
  "Input": {
    "type": "object",
    "properties": {
      "a": {
        "type": "number"
      },
      "b": {
        "type": "number"
      }
    }
  }
}

```

```
    },
    "op": {
      "type": "string"
    }
  },
  "title": "Input"
},
"Output": {
  "type": "object",
  "properties": {
    "c": {
      "type": "number"
    }
  },
  "title": "Output"
},
"Result": {
  "type": "object",
  "properties": {
    "input": {
      "$ref": "#/definitions/Input"
    },
    "output": {
      "$ref": "#/definitions/Output"
    }
  },
  "title": "Result"
}
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "SimpleCalc",
    "version" : "2016-09-29T20:27:30Z"
  },
  "servers" : [ {
    "url" : "https://t6dve4zn25.execute-api.us-west-2.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
```

```

        "default" : "demo"
    }
} ],
"paths" : {
   ("/{a}/{b}" : {
        "x-amazon-apigateway-any-method" : {
            "parameters" : [ {
                "name" : "a",
                "in" : "path",
                "required" : true,
                "schema" : {
                    "type" : "string"
                }
            }, {
                "name" : "b",
                "in" : "path",
                "required" : true,
                "schema" : {
                    "type" : "string"
                }
            } ],
            "responses" : {
                "404" : {
                    "description" : "404 response",
                    "content" : { }
                }
            },
            "x-amazon-apigateway-integration" : {
                "type" : "mock",
                "responses" : {
                    "default" : {
                        "statusCode" : "404",
                        "responseTemplates" : {
                            "application/json" : "{ \"Message\" : \"Can't $context.httpMethod $context.resourcePath\" }"
                        }
                    }
                }
            },
            "requestTemplates" : {
                "application/json" : "{ \"statusCode\" : 200 }"
            },
            "passthroughBehavior" : "when_no_match"
        }
    }
}

```

```

    }
  },
  "{a}/{b}/{op}" : {
    "get" : {
      "parameters" : [ {
        "name" : "a",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      }, {
        "name" : "b",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      }, {
        "name" : "op",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "responses" : {
        "200" : {
          "description" : "200 response",
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Result"
              }
            }
          }
        }
      }
    },
    "x-amazon-apigateway-integration" : {
      "type" : "aws",
      "httpMethod" : "POST",
      "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
      "responses" : {

```

```

        "default" : {
            "statusCode" : "200",
            "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\n\"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
\n\"$inputRoot.op\"\n },\n \"output\" : {\n    \"c\" : $inputRoot.c\n }\n}"
            }
        },
        "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\n\"a\" : $input.params('a'),\n \"b\" : $input.params('b'),\n \"op\" :
\n\"$input.params('op')\"\n}"
        },
        "passthroughBehavior" : "when_no_templates"
    }
},
"/" : {
    "get" : {
        "parameters" : [ {
            "name" : "op",
            "in" : "query",
            "schema" : {
                "type" : "string"
            }
        }, {
            "name" : "a",
            "in" : "query",
            "schema" : {
                "type" : "string"
            }
        }, {
            "name" : "b",
            "in" : "query",
            "schema" : {
                "type" : "string"
            }
        }
    ],
    "responses" : {
        "200" : {
            "description" : "200 response",
            "content" : {
                "application/json" : {

```

```

        "schema" : {
            "$ref" : "#/components/schemas/Result"
        }
    }
}
},
"x-amazon-apigateway-integration" : {
    "type" : "aws",
    "httpMethod" : "POST",
    "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
    "responses" : {
        "default" : {
            "statusCode" : "200",
            "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\\\"input\\\" : {\n    \\\"a\\\" : $inputRoot.a,\n    \\\"b\\\" : $inputRoot.b,\n    \\\"op\\\" :
\\\"$inputRoot.op\\\"}\n  },\n  \\\"output\\\" : {\n    \\\"c\\\" : $inputRoot.c\n  }\n}"
            }
        }
    },
    "requestTemplates" : {
        "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\\\"a\\\" : $input.params('a'),\n  \\\"b\\\" : $input.params('b'),\n  \\\"op\\\" :
\\\"$input.params('op')\\\"}\n}"
    },
    "passthroughBehavior" : "when_no_templates"
}
},
"post" : {
    "requestBody" : {
        "content" : {
            "application/json" : {
                "schema" : {
                    "$ref" : "#/components/schemas/Input"
                }
            }
        }
    },
    "required" : true
},
"responses" : {
    "200" : {
        "description" : "200 response",

```



```

    "content" : {
      "application/json" : {
        "schema" : {
          "$ref" : "#/components/schemas/Result"
        }
      }
    }
  },
  "x-amazon-apigateway-integration" : {
    "type" : "aws",
    "httpMethod" : "POST",
    "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
    "responses" : {
      "default" : {
        "statusCode" : "200",
        "responseTemplates" : {
          "application/json" : "#set($inputRoot = $input.path('$'))\n{\n\n  \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" : \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
        }
      }
    },
    "passthroughBehavior" : "when_no_match"
  }
},
"/{a}" : {
  "x-amazon-apigateway-any-method" : {
    "parameters" : [ {
      "name" : "a",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    } ],
    "responses" : {
      "404" : {
        "description" : "404 response",
        "content" : { }
      }
    }
  },

```

```

    "x-amazon-apigateway-integration" : {
      "type" : "mock",
      "responses" : {
        "default" : {
          "statusCode" : "404",
          "responseTemplates" : {
            "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
          }
        }
      },
      "requestTemplates" : {
        "application/json" : "{ \"statusCode\" : 200}"
      },
      "passthroughBehavior" : "when_no_match"
    }
  }
},
"components" : {
  "schemas" : {
    "Input" : {
      "title" : "Input",
      "type" : "object",
      "properties" : {
        "a" : {
          "type" : "number"
        },
        "b" : {
          "type" : "number"
        },
        "op" : {
          "type" : "string"
        }
      }
    },
    "Output" : {
      "title" : "Output",
      "type" : "object",
      "properties" : {
        "c" : {
          "type" : "number"
        }
      }
    }
  }
}

```

```
    },
    "Result" : {
      "title" : "Result",
      "type" : "object",
      "properties" : {
        "input" : {
          "$ref" : "#/components/schemas/Input"
        },
        "output" : {
          "$ref" : "#/components/schemas/Output"
        }
      }
    }
  }
}
```

Generieren des Java-SDK einer API in API Gateway

Im folgenden Verfahren wird die Generierung eines Java-SDK einer API in API Gateway demonstriert.

So generieren Sie das Java SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Java-Plattform aus und gehen Sie wie folgt vor:
 - a. Geben Sie für Service Name den Namen Ihres SDK an. Beispiel, **SimpleCalcSdk**. Dies wird der Name Ihrer SDK-Client-Klasse. Der Name entspricht dem <name>-Tag unter <project> in der pom.xml-Datei, die sich im Projektordner des SDK befindet. Verwenden Sie keine Bindestriche.

- b. Geben Sie für Java Package Name einen Paketnamen für Ihr SDK an. Beispiel, **examples.aws.apig.simpleCalc.sdk**. Dieser Paketname wird als Namespace Ihrer SDK-Bibliothek verwendet. Verwenden Sie keine Bindestriche.
 - c. Geben Sie für Java Build System **maven** oder **gradle** ein, um das Build-System anzugeben.
 - d. Geben Sie für Java Group Id einen Gruppenbezeichner für Ihr SDK-Projekt ein. Geben Sie z. B. ei **my-apig-api-examples**. Dieser Bezeichner entspricht dem <groupId>-Tag unter <project> in der pom.xml-Datei, die sich im Projektordner des SDK befindet.
 - e. Geben Sie für Java Artifact Id einen Artefaktbezeichner für Ihr SDK-Projekt ein. Geben Sie z. B. ei **simple-calc-sdk**. Dieser Bezeichner entspricht dem <artifactId>-Tag unter <project> in der pom.xml-Datei, die sich im Projektordner des SDK befindet.
 - f. Geben Sie für Java Artifact Version eine Versions-ID-Zeichenfolge ein. Beispiel, **1.0.0**. Diese Version entspricht dem <version>-Tag unter <project> in der pom.xml-Datei, die sich im Projektordner des SDK befindet.
 - g. Geben Sie für Source Code License Text den Lizenztext Ihres Quellcodes ein, falls vorhanden.
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in [Von API Gateway generiertes Java-SDK für eine REST-API verwenden](#) zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des Android SDK einer API in API Gateway

Im folgenden Verfahren wird die Generierung eines Android SDK einer API in API Gateway demonstriert.

So generieren Sie das Android SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.

4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Android-Plattform aus und gehen Sie wie folgt vor:
 - a. Geben Sie für Group ID den eindeutigen Bezeichner für das entsprechende Projekt ein. Dieser wird in der `pom.xml`-Datei verwendet (zum Beispiel **`com.mycompany`**).
 - b. Geben Sie für Invoker package den Namespace für die generierten Client-Klassen ein (zum Beispiel **`com.mycompany.clientsdk`**).
 - c. Geben Sie für Artifact ID den Namen der kompilierten JAR-Datei ohne die Version ein. Dieser wird in der `pom.xml`-Datei verwendet (zum Beispiel **`aws-apigateway-api-sdk`**).
 - d. Geben Sie für Artifact version die Artefakt-Versionsnummer für den generierten Client ein. Dies wird in der `pom.xml` Datei verwendet und sollte einem folgen. *major minor.patch* Muster (zum Beispiel **`1.0.0`**).
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in [Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden](#) zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des iOS SDK einer API in API Gateway

Im folgenden Verfahren wird die Generierung eines iOS SDK einer API in API Gateway demonstriert.

So generieren Sie das iOS SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie für Plattform die Plattform iOS (Objective-C) oder iOS (Swift) aus und gehen Sie wie folgt vor:

- Geben Sie für im Feld Prefix ein eindeutiges Präfix ein.

Das Präfix hat folgende Wirkung: Wenn Sie beispielsweise als Präfix für das SDK der API `result` Modelle mit, **SIMPLE_CALC** und zuweisen `input`, `output` enthält das generierte SDK die `SIMPLE_CALCSimpleCalcClient` Klasse, die die [SimpleCalcAPI](#) kapselt, einschließlich der Methodenanforderungen/-antworten. Darüber hinaus enthält das generierte SDK die `SIMPLE_CALCinput`-, `SIMPLE_CALCoutput`- und `SIMPLE_CALCresult`-Klassen, um Eingabe, Ausgabe und Ergebnisse bzw. die Anforderungseingabe und Antwortausgabe zu repräsentieren. Weitere Informationen finden Sie unter [Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden](#).

7. Wählen Sie **Generate SDK** (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in [Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden](#) zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren Sie das JavaScript SDK einer REST-API in API Gateway

Das folgende Verfahren zeigt, wie das JavaScript SDK einer API in API Gateway generiert wird.

Um das JavaScript SDK einer API in API Gateway zu generieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann **SDK generieren** aus.
6. Wählen Sie unter Plattform die Plattform aus. **JavaScript**
7. Wählen Sie **Generate SDK** (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in [Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API](#) zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des Ruby SDK einer API in API Gateway

Im folgenden Verfahren wird die Generierung eines Ruby SDK einer API in API Gateway demonstriert.

So generieren Sie das Ruby SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Ruby-Plattform aus und gehen Sie wie folgt vor:
 - a. Geben Sie für Service Name den Namen Ihres SDK an. Beispiel, **SimpleCalc**. Dieser Name wird verwendet, um den Ruby Gem Namespace Ihrer API zu generieren. Der Name besteht nur aus Buchstaben (a-zA-Z) ohne Sonderzeichen oder Zahlen.
 - b. Geben Sie im Feld Ruby Gem Name den Namen des Ruby Gem ein, das den generierten SDK-Quellcode für Ihre API enthalten soll. Standardmäßig ist dies der Servicename in Kleinbuchstaben plus -sdk-Suffix, z. B. **simplecalc-sdk**.
 - c. Geben Sie für Ruby Gem Version eine Versionsnummer für das generierte Ruby Gem ein. Standardmäßig ist der Wert eingestellt 1.0.0.
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in [Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden](#) zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren Sie SDKs für eine API mithilfe von AWS CLI Befehlen in API Gateway

Sie können AWS CLI es verwenden, um ein SDK einer API für eine unterstützte Plattform zu generieren und herunterzuladen, indem Sie den Befehl [get-sdk](#) aufrufen. Im folgenden Abschnitt wird dieser Vorgang für einige der unterstützten Plattformen beschrieben.

Themen

- [Generieren Sie das SDK für Java für Android und laden Sie es herunter mit dem AWS CLI](#)
- [Generieren Sie das JavaScript SDK und laden Sie es herunter mit dem AWS CLI](#)
- [Generieren Sie das Ruby-SDK und laden Sie es herunter mit dem AWS CLI](#)

Generieren Sie das SDK für Java für Android und laden Sie es herunter mit dem AWS CLI

Zum Erstellen und Herunterladen eines von API Gateway generierten Java für Android SDKs einer API (udpuvvzbkc) auf einer bestimmten Stufe (test) rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
    --rest-api-id udpuvvzbkc \
    --stage-name test \
    --sdk-type android \
    --parameters groupId='com.mycompany',\
        invokerPackage='com.mycompany.myApiSdk',\
        artifactId='myApiSdk',\
        artifactVersion='0.0.1' \
    ~/apps/myApi/myApi-android-sdk.zip
```

Die letzte Eingabe von `~/apps/myApi/myApi-android-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-android-sdk.zip`.

Generieren Sie das JavaScript SDK und laden Sie es herunter mit dem AWS CLI

Um ein vom API Gateway generiertes JavaScript SDK einer API () zu einem bestimmten Zeitpunkt (udpuvvzbkctest) zu generieren und herunterzuladen, rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
    --rest-api-id udpuvvzbkc \
    --stage-name test \
    --sdk-type javascript \
    ~/apps/myApi/myApi-js-sdk.zip
```


Die letzte Eingabe von `~/apps/myApi/myApi-js-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-js-sdk.zip`.

Generieren Sie das Ruby-SDK und laden Sie es herunter mit dem AWS CLI

Zum Erstellen und Herunterladen eines Ruby SDKs einer API (`udpuvvzbkc`) auf einer bestimmten Stufe (`test`) rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
    --rest-api-id udpuvvzbkc \
    --stage-name test \
    --sdk-type ruby \
    --parameters service.name=myApiRubySdk,ruby.gem-name=myApi,ruby.gem-
version=0.01 \
    ~/apps/myApi/myApi-ruby-sdk.zip
```

Die letzte Eingabe von `~/apps/myApi/myApi-ruby-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-ruby-sdk.zip`.

Als Nächstes zeigen wir, wie Sie mit dem generierten SDK die zugrunde liegende API aufrufen. Weitere Informationen finden Sie unter [Aufrufen von REST-APIs in API Gateway](#).

Verkaufen Sie Ihr API Gateway APIs über AWS Marketplace

Nachdem Sie Ihre erstellt, getestet und bereitgestellt haben APIs, können Sie sie in einem API Gateway [Gateway-Nutzungsplan](#) verpacken und den Plan als SaaS-Produkt (Software as a Service) verkaufen AWS Marketplace. API-Käufern, die Ihr Produktangebot abonnieren, wird auf der AWS Marketplace Grundlage der Anzahl der Anfragen an den Nutzungsplan abgerechnet.

Um Ihr eigenes APIs zu verkaufen AWS Marketplace, müssen Sie den Vertriebskanal für die Integration AWS Marketplace mit API Gateway einrichten. Im Allgemeinen beinhaltet dies, dass Sie Ihr Produkt anbieten AWS Marketplace, eine IAM-Rolle mit entsprechenden Richtlinien einrichten, damit API Gateway Nutzungsdaten an diese senden kann AWS Marketplace, ein AWS Marketplace Produkt einem API Gateway Gateway-Nutzungsplan zuordnen und einem AWS Marketplace Käufer einen API-Gateway-API-Schlüssel zuordnen. Details dazu finden Sie in den folgenden Abschnitten.

Weitere Informationen zum Verkauf Ihrer API als SaaS-Produkt finden Sie im [AWS Marketplace Benutzerhandbuch](#). AWS Marketplace

Themen

- [Initialisieren Sie AWS Marketplace die Integration mit API Gateway](#)
- [Verarbeiten von Kundenabonnements eines Nutzungsplans](#)

Initialisieren Sie AWS Marketplace die Integration mit API Gateway

Die folgenden Aufgaben betreffen die einmalige Initialisierung der AWS Marketplace Integration mit API Gateway, sodass Sie Ihr Produkt APIs als SaaS verkaufen können.

Bieten Sie ein Produkt auf AWS Marketplace

Übermitteln Sie für die Eintragung Ihres Nutzungsplans als SaaS-Produkt ein Produktladeformular über [AWS Marketplace](#). Das Produkt muss eine Dimension mit dem Namen `apigateway` vom Typ `requests` enthalten. Diese Dimension definiert die `price-per-request` und wird von API Gateway verwendet, um Anfragen an Ihre zu messen APIs.

Erstellen der Messungsrolle

Erstellen Sie eine IAM-Rolle mit dem Namen `ApiGatewayMarketplaceMeteringRole` und der folgenden Ausführungsrichtlinie und Vertrauensrichtlinie. Diese Rolle ermöglicht es API Gateway, in Ihrem Namen Nutzungsmetriken AWS Marketplace an zu senden.

Ausführungsrichtlinie der Messungsrolle

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Vertrauensbeziehungsrichtlinie der Messungsrolle

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Verknüpfen Sie den Nutzungsplan mit dem AWS Marketplace Produkt

Wenn Sie ein Produkt anbieten AWS Marketplace, erhalten Sie einen AWS Marketplace Produktcode. Um API Gateway zu integrieren AWS Marketplace, verknüpfen Sie Ihren Nutzungsplan mit dem AWS Marketplace Produktcode. Sie aktivieren die Zuordnung, indem Sie das [productCode](#) Feld UsagePlan des API-Gateways auf Ihren AWS Marketplace Produktcode setzen, indem Sie die API Gateway-Konsole, die API Gateway Gateway-REST-API, die AWS CLI für API Gateway oder das AWS SDK für API Gateway verwenden. Das folgende Codebeispiel verwendet die API Gateway-REST-API:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "patchOperations" : [{
    "path" : "/productCode",
    "value" : "MARKETPLACE_PRODUCT_CODE",
    "op" : "replace"
  }]
}
```

Verarbeiten von Kundenabonnements eines Nutzungsplans

Die folgenden Aufgaben werden von Ihrer Entwicklerportalanwendung übernommen.

Wenn ein Kunde Ihr Produkt abonniert AWS Marketplace, AWS Marketplace leitet eine POST Anfrage an die SaaS-Abonnement-URL weiter, unter der Sie Ihr Produkt angeboten haben. AWS Marketplace Die POST-Anforderung enthält einen `x-amzn-marketplace-token`-Parameter mit Käuferinformationen. Befolgen Sie die Anweisungen unter [Onboarding für SaaS-Kunden](#), um diese Umleitung in Ihrer Entwicklerportalanwendung zu verarbeiten.

Wenn Sie auf die Abonnementanfrage eines Kunden antworten, wird eine `subscribe-success` Benachrichtigung zu einem Amazon SNS SNS-Thema AWS Marketplace gesendet, das Sie abonnieren können. (Informationen finden Sie unter [Onboarding für SaaS-Kunden](#)). Um die Abonnementanfrage des Kunden anzunehmen, bearbeiten Sie die `subscribe-success` Benachrichtigung, indem Sie einen AWS Marketplace API-Gateway-API-Schlüssel für den Kunden erstellen oder abrufen, den vom Kunden bereitgestellten API-Schlüssel `customerId` mit den API-Schlüsseln verknüpfen und den API-Schlüssel dann zu Ihrem Nutzungsplan hinzufügen.

Wenn die Abonnementanfrage des Kunden abgeschlossen ist, sollte die Entwicklerportal-Anwendung dem Kunden den zugehörigen API-Schlüssel vorlegen und den Kunden darüber informieren, dass der API-Schlüssel in der `x-api-key` Kopfzeile von Anfragen an den enthalten sein muss. APIs

Wenn ein Kunde ein Abonnement für einen Nutzungsplan kündigt, AWS Marketplace sendet er eine `unsubscribe-success` Benachrichtigung zum Thema SNS. Zum Abschließen der Abonnementkündigung des Kunden verarbeiten Sie die `unsubscribe-success`-Benachrichtigung, indem Sie die API-Schlüssel des Kunden aus dem Nutzungsplan entfernen.

Autorisieren eines Kunden für den Zugriff auf den Nutzungsplan

Verwenden Sie zum Autorisieren des Zugriffs eines Kunden auf Ihren Nutzungsplan die API Gateway-API, um einen API-Schlüssel abzurufen oder zu erstellen, und fügen Sie dem Nutzungsplan den API-Schlüssel hinzu.

Das folgende Beispiel zeigt, wie die API Gateway REST API aufgerufen wird, um einen neuen API-Schlüssel mit einem bestimmten AWS Marketplace `customerId` Wert (*MARKETPLACE_CUSTOMER_ID*) zu erstellen.

```
POST apikeys HTTP/1.1
Host: apigateway.region.amazonaws.com
```

```
Authorization: ...

{
  "name" : "my_api_key",
  "description" : "My API key",
  "enabled" : "false",
  "stageKeys" : [ {
    "restApiId" : "uycll6xg9a",
    "stageName" : "prod"
  } ],
  "customerId" : "MARKETPLACE_CUSTOMER_ID"
}
```

Das folgende Beispiel zeigt, wie Sie einen API-Schlüssel mit einem bestimmten AWS Marketplace `customerId` Wert (`MARKETPLACE_CUSTOMER_ID`) abrufen.

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Erstellen Sie zum Hinzufügen eines API-Schlüssels zu einem Nutzungsplan einen [UsagePlanKey](#) mit dem API-Schlüssel für den entsprechenden Nutzungsplan. Im folgenden Beispiel wird gezeigt, wie dies über die API Gateway-REST-API durchgeführt wird, wobei `n371pt` die Nutzungsplan-ID und `q5ugs7qjjh` eine Beispiel-API-keyId ist, die in den vorhergehenden Beispielen zurückgegeben wurde.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "keyId": "q5ugs7qjjh",
  "keyType": "API_KEY"
}
```

Verknüpfen eines Kunden mit einem API-Schlüssel

Sie müssen [ApiKey](#) das `customerId` Feld mit der AWS Marketplace Kunden-ID des Kunden aktualisieren. Auf diese Weise wird der API-Schlüssel mit dem AWS Marketplace -Kunden verknüpft, was die Messung und Abrechnung dieses Käufers ermöglicht. Im folgenden Codebeispiel wird die API Gateway-REST-API aufgerufen, um dies durchzuführen.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "patchOperations" : [{
    "path" : "/customerId",
    "value" : "MARKETPLACE_CUSTOMER_ID",
    "op" : "replace"
  }]
}
```

Schützen Sie Ihren REST APIs in API Gateway

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, wie böswillige Benutzer oder Spitzen im Datenverkehr. Sie können Ihre API mit Strategien schützen, wie das Generieren von SSL-Zertifikaten, das Konfigurieren einer Firewall für Webanwendungen, das Festlegen von Drosselungs-Targets und das Ermöglichen des Zugriffs auf Ihre API über eine Virtual Private Cloud (VPC). In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen mit API Gateway aktivieren können.

Themen

- [So aktivieren Sie die gegenseitige TLS-Authentifizierung für Ihren REST APIs in API Gateway](#)
- [Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-Authentifizierung in API Gateway](#)
- [Verwenden Sie AWS WAF , um Ihren REST APIs in API Gateway zu schützen](#)
- [Drosseln Sie Anfragen an Ihren REST APIs für einen besseren Durchsatz in API Gateway](#)
- [Privates REST APIs im API Gateway](#)

So aktivieren Sie die gegenseitige TLS-Authentifizierung für Ihren REST APIs in API Gateway

Die gegenseitige TLS-Authentifizierung erfordert eine bidirektionale Authentifizierung zwischen dem Client und dem Server. Bei gegenseitiger TLS müssen Clients X.509-Zertifikate zur Prüfung ihrer Identität präsentieren, um auf Ihre API zuzugreifen. Gegenseitiges TLS ist eine allgemeine Anforderung für das Internet der Dinge (IoT) und business-to-business Anwendungen.

Sie können gegenseitiges TLS zusammen mit anderen [Autorisierungs- und Authentifizierungsvorgängen](#) verwenden, die API Gateway unterstützt. API Gateway leitet die von Kunden bereitgestellten Zertifikate an Lambda-Genehmiger und an Backend-Integrationen weiter.

 Important

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Clients nur mithilfe eines benutzerdefinierten Domänennamens mit gegenseitiger TLS auf Ihre API zugreifen können, deaktivieren Sie den `execute-api`-Standardendpunkt. Weitere Informationen hierzu finden Sie unter [the section called “Deaktivieren des Standardendpunkts”](#).

Themen

- [Voraussetzungen für gegenseitige TLS](#)
- [Konfigurieren von gegenseitiger TLS für einen benutzerdefinierten Domänennamen](#)
- [Aufrufen einer API mithilfe eines benutzerdefinierten Domänennamens, der gegenseitige TLS erfordert](#)
- [Aktualisieren Ihres Truststore](#)
- [Gegenseitige TLS deaktivieren](#)
- [Problembehebung bei gegenseitigen TLS für Ihre REST-API](#)

Voraussetzungen für gegenseitige TLS

Um gegenseitige TLS zu konfigurieren, benötigen Sie:

- Einen regionalen benutzerdefinierten Domainnamen
- AWS Certificate Manager Für Ihren benutzerdefinierten Domainnamen ist mindestens ein Zertifikat konfiguriert
- Ein in Amazon S3 konfigurierter und hochgeladener Truststore

Benutzerdefinierte Domainnamen

Um die gegenseitige TLS-Authentifizierung für eine REST-API zu aktivieren, müssen Sie einen benutzerdefinierten Domänennamen für Ihre API konfigurieren. Sie können gegenseitige TLS

für einen benutzerdefinierten Domännennamen aktivieren und dann den benutzerdefinierten Domännennamen für Clients bereitstellen. Um mithilfe eines benutzerdefinierten Domännennamens auf eine API zuzugreifen, für die gegenseitige TLS aktiviert ist, müssen Clients Zertifikate präsentieren, denen Sie in API-Anforderungen vertrauen. Sie finden mehr Informationen in [the section called “Benutzerdefinierte Domännennamen”](#).

Verwendung AWS Certificate Manager ausgestellter Zertifikate

Sie können ein öffentlich vertrauenswürdiges Zertifikat direkt von ACM anfordern oder öffentliche oder selbstsignierte Zertifikate importieren. Gehen Sie zum Einrichten eines Zertifikats in ACM zu [ACM](#). Wenn Sie ein Zertifikat importieren möchten, lesen Sie weiter im folgenden Abschnitt.

Verwenden eines importierten AWS Private Certificate Authority Zertifikats oder Zertifikats

Um ein in ACM importiertes Zertifikat oder ein Zertifikat von Mutual TLS verwenden AWS Private Certificate Authority zu können, benötigt API Gateway ein von ACM `ownershipVerificationCertificate` ausgestelltes. Dieses Eigentumszertifikat wird nur verwendet, um zu überprüfen, ob Sie über Berechtigungen zur Verwendung des Domännennamens verfügen. Es wird nicht für den TLS-Handshake verwendet. Falls Sie noch keinen `ownershipVerificationCertificate` haben, gehen Sie zu, <https://console.aws.amazon.com/acm/> um einen einzurichten.

Sie müssen dieses Zertifikat für die gesamte Lebensdauer Ihres Domainnamens gültig halten. Wenn ein Zertifikat abläuft und die automatische Verlängerung fehlschlägt, werden alle Updates des Domännennamens gesperrt. Sie müssen die `ownershipVerificationCertificateArn` mit einem gültigen `ownershipVerificationCertificate` aktualisieren, bevor Sie weitere Änderungen vornehmen können. Die `ownershipVerificationCertificate` kann nicht als Serverzertifikat für eine andere gemeinsame TLS-Domäne in API Gateway verwendet werden. Wenn ein Zertifikat direkt wieder in ACM importiert wird, muss der Aussteller gleich bleiben.

Konfigurieren Ihres Truststore

Truststores sind Textdateien mit einer `.pem`-Dateierweiterung. Sie sind eine vertrauenswürdige Liste von Zertifikaten von Zertifizierungsstellen. Um Mutual TLS zu verwenden, erstellen Sie einen Truststore mit X.509-Zertifikaten, denen Sie beim Zugriff auf Ihre API vertrauen.

Sie müssen die vollständige Vertrauenskette vom ausstellenden CA-Zertifikat bis zum Stamm-CA-Zertifikat in Ihren Truststore aufnehmen. API Gateway akzeptiert Clientzertifikate, die von jeder Zertifizierungsstelle in der Vertrauenskette ausgestellt werden. Die Zertifikate können von öffentlichen oder privaten Zertifizierungsstellen stammen. Zertifikate können eine maximale Kettenlänge von vier

haben. Sie können auch selbstsignierte Zertifikate bereitstellen. Die folgenden Algorithmen werden im Truststore unterstützt:

- SHA-256 oder stärker
- RSA-2048 oder stärker
- ECDSA-256 oder ECDSA-384

API Gateway validiert eine Reihe von Zertifikatseigenschaften. Sie können Lambda-Generierer verwenden, um zusätzliche Prüfungen durchzuführen, wenn ein Client eine API aufruft, einschließlich der Prüfung, ob ein Zertifikat widerrufen wurde. API Gateway validiert die folgenden Eigenschaften:

Validierung	Beschreibung
X.509-Syntax	Das Zertifikat muss die X.509-Syntaxanforderungen erfüllen.
Integrität	Der Inhalt des Zertifikats darf sich nicht von dem unterscheiden, was von der Zertifizierungsstelle des Truststore signiert wurde.
Gültigkeit	Die Gültigkeitsdauer des Zertifikats muss aktuell sein.
Namensverkettung/Schlüsselverkettung	Die Verkettung der Namen und Antragssteller von Zertifikaten darf nicht unterbrochen sein. Zertifikate können eine maximale Kettenlänge von vier haben.

Hochladen des Vertrauensspeichers in einen Amazon-S3-Bucket in einer einzigen Datei.

Im Folgenden finden Sie ein Beispiel dafür, wie eine .pem-Datei aussehen könnte.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----
```

```
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Der folgende [AWS CLI cp-Befehl](#) lädt `certificates.pem` in Ihren Amazon S3 S3-Bucket hoch:

```
aws s3 cp certificates.pem s3://bucket-name
```

Konfigurieren von gegenseitiger TLS für einen benutzerdefinierten Domännennamen

Sie müssen einen benutzerdefinierten regionalen Domainnamen mit einer TLS_1_2-Sicherheitsrichtlinie für Ihre API verwenden, um gegenseitiges TLS für eine REST-API zu konfigurieren. Weitere Informationen zur Auswahl einer Sicherheitsrichtlinie finden Sie unter [the section called “Eine Sicherheitsrichtlinie auswählen”](#).

Note

Mutual TLS wird für private APIs Zwecke nicht unterstützt.

Nachdem Sie Ihren Vertrauensspeicher in Amazon S3 hochgeladen haben, können Sie Ihren benutzerdefinierten Domännennamen für die Verwendung von gegenseitigem TLS konfigurieren. Im Folgenden [create-domain-name](#) wird ein benutzerdefinierter Domainname mit gegenseitigem TLS erstellt:

```
aws apigateway create-domain-name --region us-east-2 \
  --domain-name api.example.com \
  --regional-certificate-arn arn:aws:acm:us-east-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
  --endpoint-configuration types=REGIONAL \
  --security-policy TLS_1_2 \
  --mutual-tls-authentication truststoreUri=s3://bucket-name/key-name
```

Nachdem Sie den Domännennamen erstellt haben, müssen Sie DNS-Datensätze und Basepath-Mappings für API-Vorgänge konfigurieren. Weitere Informationen hierzu finden Sie unter [Einrichten eines regionalen benutzerdefinierten Domainnamens in API Gateway](#).

Aufrufen einer API mithilfe eines benutzerdefinierten Domänennamens, der gegenseitige TLS erfordert

Um eine API mit aktivierter gegenseitiger TLS aufzurufen, müssen Clients in der API-Anforderung ein vertrauenswürdiges Zertifikat präsentieren. Wenn ein Client versucht, Ihre API aufzurufen, sucht API Gateway in Ihrem Truststore nach dem Aussteller des Clientzertifikats. Damit API Gateway mit der Anforderung fortfahren kann, müssen sich der Aussteller des Zertifikats und die vollständige Vertrauenskette bis zum Stamm-CA-Zertifikats in Ihrem Truststore befinden.

Mit dem folgenden `curl`-Beispielbefehl wird eine Anforderung an `api.example.com`, gesendet, die `my-cert.pem` in die Anforderung einschließt. `my-key.key` ist der private Schlüssel für das Zertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Ihre API wird nur aufgerufen, wenn Ihr Truststore dem Zertifikat vertraut. Die folgenden Bedingungen führen dazu, dass API Gateway den TLS-Handshake fehlschlägt und die Anforderung mit einem 403-Statuscode verweigert. Wenn Ihr Zertifikat:

- nicht vertrauenswürdig ist
- abgelaufen ist
- keinen unterstützten Algorithmus verwendet

Note

API Gateway überprüft nicht, ob ein Zertifikat widerrufen wurde.

Aktualisieren Ihres Truststore

Um die Zertifikate in Ihrem Truststore zu aktualisieren, laden Sie ein neues Zertifikatspaket in Amazon S3 hoch. Anschließend können Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um das aktualisierte Zertifikat zu verwenden.

Verwenden Sie [Amazon-S3-Versioning](#), um mehrere Versionen Ihres Vertrauensspeichers zu verwalten. Wenn Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um eine neue Vertrauensspeicher-Version zu verwenden, gibt API Gateway Warnungen zurück, wenn Zertifikate ungültig sind.

API Gateway erzeugt Zertifikatswarnungen nur dann, wenn Sie Ihren Domännennamen aktualisieren. API Gateway benachrichtigt Sie nicht, wenn ein zuvor hochgeladenes Zertifikat abläuft.

Mit dem folgenden [update-domain-name](#) Befehl wird ein benutzerdefinierter Domainname aktualisiert, sodass er eine neue Truststore-Version verwendet:

```
aws apigateway update-domain-name \  
  --domain-name api.example.com \  
  --patch-operations op='replace',path='/mutualTlsAuthentication/  
truststoreVersion',value='abcdef123'
```

Gegenseitige TLS deaktivieren

Mit dem Folgenden wird Mutual [update-domain-name](#) TLS deaktiviert:

```
aws apigateway update-domain-name \  
  --domain-name api.example.com \  
  --patch-operations op='replace',path='/mutualTlsAuthentication/  
truststoreUri',value=''
```

Problembehebung bei gegenseitigen TLS für Ihre REST-API

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Aktivierung von „Gegenseitige TLS“ auftreten können.

Fehlerbehebung bei Zertifikatswarnungen

Wenn Sie einen benutzerdefinierten Domännennamen mit gegenseitiger TLS erstellen, gibt API Gateway Warnungen zurück, wenn Zertifikate im Truststore ungültig sind. Dies kann auch auftreten, wenn ein benutzerdefinierter Domänenname aktualisiert wird, um einen neuen Truststore zu verwenden. Die Warnungen geben das Problem mit dem Zertifikat und den Betreff des Zertifikats an, das die Warnung erstellt hat. Gegenseitige TLS ist weiterhin für Ihre API aktiviert, aber einige Clients können möglicherweise nicht auf Ihre API zugreifen.

Um aufzudecken, welches Zertifikat die Warnung erzeugt hat, dekodieren Sie die Zertifikate in Ihrem Truststore. Sie können Tools wie `openssl` zum Dekodieren der Zertifikate und zur Identifizierung ihrer Subjekte verwenden.

Der folgende Befehl zeigt den Inhalt eines Zertifikats einschließlich seines Betreffs an:

```
openssl x509 -in certificate.crt -text -noout
```

Aktualisieren oder entfernen Sie die Zertifikate, die Warnungen erstellt haben, und laden Sie dann einen neuen Truststore in Amazon S3 hoch. Nachdem Sie einen neuen Truststore hochgeladen haben, aktualisieren Sie Ihren benutzerdefinierten Domännennamen, um den neuen Truststore zu verwenden.

Problembehandlung bei Domännennamenskonflikten

Der Fehler "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." bedeutet, dass mehrere Zertifizierungsstellen ein Zertifikat für diese Domäne ausgestellt haben. Für jedes Subjekt im Zertifikat kann es nur einen Aussteller in API Gateway für gegenseitige TLS-Domänen geben. Sie müssen alle Ihre Zertifikate für dieses Thema über einen einzigen Aussteller abrufen. Wenn das Problem mit einem Zertifikat besteht, das Sie nicht kontrollieren können, Sie jedoch den Besitz des Domännennamens nachweisen können, [kontaktieren Sie Support](#), um ein Ticket zu eröffnen.

Fehlerbehebung bei Statusmeldungen für Domännennamen

PENDING_CERTIFICATE_REIMPORT: Das bedeutet, dass Sie ein Zertifikat erneut in ACM importiert haben und die Validierung fehlgeschlagen ist, weil das neue Zertifikat ein SAN (Subject Alternative Name) hat, das nicht durch den `ownershipVerificationCertificate` oder der Betreff abgedeckt ist oder der Betreff oder SANs im Zertifikat nicht den Domainnamen abdeckt. Möglicherweise ist etwas falsch konfiguriert oder ein ungültiges Zertifikat wurde importiert. Sie müssen ein gültiges Zertifikat erneut in ACM importieren. Weitere Informationen zur Validierung finden Sie unter [Validierung des Domänenbesitzes](#).

PENDING_OWNERSHIP_VERIFICATION: Dies bedeutet, dass Ihr zuvor verifiziertes Zertifikat abgelaufen ist und ACM es nicht automatisch erneuern konnte. Sie müssen das Zertifikat erneuern oder ein neues Zertifikat anfordern. Weitere Informationen zur Zertifikatserneuerung finden Sie im Leitfaden [ACMs Fehlerbehebung bei der Erneuerung verwalteter Zertifikate](#).

Fehlerbehebung bei einem inkorrekt zurückgegebene Zertifikat

Bei der Migration eines dedizierten Zertifikats von einem vollqualifizierten Domainnamen (FQDN) zu einem Platzhalter-Kundendomainnamen gibt API Gateway möglicherweise das Zertifikat für den FQDN und nicht für den Platzhalter-Domainnamen zurück.

Im folgenden Befehl wird gezeigt, welches Zertifikat von API Gateway zurückgegeben wird:

```
openssl s_client -connect hostname:port
```

Wenn das resultierende Zertifikat für den FQDN ist, [wenden Sie sich an Support](#), um ein Ticket zu eröffnen.

Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-Authentifizierung in API Gateway

Sie können API Gateway verwenden, um ein SSL-Zertifikat zu generieren und dann dessen öffentlichen Schlüssel im Backend verwendet, um zu überprüfen, ob HTTP-API-Anfragen an Ihr Backend-System von API Gateway stammen. So kann Ihr HTTP-Backend nur Anfragen steuern und akzeptieren, die von Amazon API Gateway stammen – selbst wenn das Backend öffentlich zugänglich ist.

Note

Einige Backend-Server unterstützen möglicherweise keine SSL-Client-Authentifizierung als API Gateway. Sie könnten einen SSL-Zertifikatsfehler zurückgeben. Eine Liste der inkompatiblen Backend-Server finden Sie unter [the section called “Wichtige Hinweise”](#).

Die von API Gateway generierten SSL-Zertifikate sind selbstsigniert, und nur der öffentliche Schlüssel eines Zertifikats ist in der API Gateway Gateway-Konsole oder über die APIs sichtbar.

Themen

- [Client-Zertifikat über die API Gateway-Konsole generieren](#)
- [Konfigurieren einer API für die Verwendung von SSL-Zertifikaten](#)
- [Testaufruf zur Überprüfung der Konfiguration des Clientzertifikats](#)
- [Konfigurieren eines Backend-HTTPS-Servers zur Überprüfung des Clientzertifikats](#)
- [Rotieren ablaufender Clientzertifikate](#)
- [Von API Gateway unterstützte Zertifizierungsstellen für HTTP- und HTTP-Proxy-Integrationen](#)

Client-Zertifikat über die API Gateway-Konsole generieren

1. Öffnen Sie die API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway/>.
2. Wählen Sie einen REST oder eine WebSocket API.

3. Wählen Sie im Hauptnavigationsfenster Client Certificates (Clientzertifikate).
4. Wählen Sie auf der Seite Client Certificates (Clientzertifikate) die Option Generate certificate (Zertifikat generieren) aus.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
6. Wählen Sie Generate certificate (Zertifikat generieren) aus, um das Zertifikat zu generieren. API Gateway generiert ein neues Zertifikat und gibt die neue Zertifikats-GUID zusammen mit dem PEM-kodierten öffentlichen Schlüssel zurück.

Sie können eine API nun für die Verwendung des Zertifikats konfigurieren.

Konfigurieren einer API für die Verwendung von SSL-Zertifikaten

Bei diesen Anleitungen wird davon ausgegangen, dass Sie bereits abgeschlossen haben [Client-Zertifikat über die API Gateway-Konsole generieren](#).

1. Erstellen oder öffnen Sie in der API Gateway Gateway-Konsole einen REST oder eine WebSocket API, für die Sie das Client-Zertifikat verwenden möchten. Stellen Sie sicher, dass die API für eine Stufe bereitgestellt wurde.
2. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
3. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Client certificate (Clientzertifikat) ein Zertifikat aus.
5. Wählen Sie Änderungen speichern aus.

Nachdem ein Zertifikat für die API ausgewählt und gespeichert wurde, verwendet API Gateway das Zertifikat für alle Aufrufe an HTTP-API-Integrationen in Ihrer API.

Testaufruf zur Überprüfung der Konfiguration des Clientzertifikats

1. Wählen Sie eine REST-API-Methode aus. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Test anzuzeigen.
2. Wählen Sie für Client certificate (Clientzertifikat) ein Zertifikat aus.
3. Wählen Sie Test aus.

API Gateway nutzt das ausgewählte SSL-Zertifikat für das HTTP-Backend, um die API zu authentifizieren.

Konfigurieren eines Backend-HTTPS-Servers zur Überprüfung des Clientzertifikats

Diese Anleitungen setzen voraus, dass Sie die in [Client-Zertifikat über die API Gateway-Konsole generieren](#) beschriebenen Aktionen ausgeführt und eine Kopie des Clientzertifikats heruntergeladen haben. Sie können ein Client-Zertifikat herunterladen, indem Sie [clientcertificate:by-id](#) der API-Gateway-REST-API oder [get-client-certificate](#) der AWS CLI aufrufen.

Vor der Konfiguration eines Backend-HTTPS-Servers zur Überprüfung des SSL-Clientzertifikats von Amazon API Gateway müssen Sie den PEM-kodierten privaten Schlüssel und ein von einer vertrauenswürdigen Zertifizierungsstelle bereitgestelltes serverseitiges Zertifikat erhalten haben.

Wenn der Name der Serverdomäne `myserver.mydomain.com` ist, muss der CNAME-Wert des Serverzertifikats `myserver.mydomain.com` oder `*.mydomain.com` sein.

Zu den unterstützten Zertifizierungsstellen gehören [Let's Encrypt](#) oder eine der in [the section called "Unterstützte Zertifizierungsstellen für die HTTP- und HTTP-Proxy-Integration"](#) aufgelisteten Stellen.

Angenommen, die Clientzertifikatdatei ist `apig-cert.pem` und der private Schlüssel und die Zertifikatdateien des Servers sind `server-key.pem` bzw. `server-cert.pem`. Für einen Node.js-Server im Backend können Sie den Server wie folgt konfigurieren:

```
var fs = require('fs');
var https = require('https');
var options = {
  key: fs.readFileSync('server-key.pem'),
  cert: fs.readFileSync('server-cert.pem'),
  ca: fs.readFileSync('apig-cert.pem'),
  requestCert: true,
  rejectUnauthorized: true
};
https.createServer(options, function (req, res) {
  res.writeHead(200);
  res.end("hello world\n");
}).listen(443);
```

Für eine [Node-Express-App](#) können Sie die [client-certificate-auth](#) Module verwenden, um Client-Anfragen mit PEM-codierten Zertifikaten zu authentifizieren.

Informationen zu anderen HTTPS-Servers finden Sie in der Dokumentation zum Server.

Rotieren ablaufender Clientzertifikate

Das von API Gateway generierte Client-Zertifikat ist 365 Tage gültig. Sie müssen das Zertifikat rotieren, bevor ein Clientzertifikat auf einer API-Stufe abläuft, um Ausfallzeiten für die API zu vermeiden.

Wechseln Sie ein ablaufendes Client-Zertifikat mithilfe der AWS-Managementkonsole

Das folgende Verfahren zeigt, wie Sie ein Clientzertifikat in der Konsole für eine zuvor bereitgestellte API rotieren.

1. Wählen Sie im Hauptnavigationsfenster Client Certificates (Clientzertifikate).
2. Wählen Sie im Bereich Client Certificates ((Clientzertifikate) Generate certificate (Zertifikat generieren) aus.
3. Öffnen Sie die API, für die Sie das Client-Zertifikat verwenden möchten.
4. Wählen Sie unter der ausgewählten API Stages und wählen Sie dann eine Stufe aus.
5. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
6. Wählen Sie für Client certificate (Clientzertifikat) das neue Zertifikat aus.
7. Wählen Sie zum Speichern der Einstellungen Save changes (Änderungen speichern).

Rotieren Sie ein ablaufendes Client-Zertifikat mithilfe der AWS CLI

[Sie können das Ablaufdatum des Zertifikats überprüfen, indem Sie clientCertificate:by-ID der API Gateway Gateway-REST-API aufrufen oder den AWS CLI Befehl von aufrufen get-client-certificate und die zurückgegebene Eigenschaft expirationDate überprüfen.](#)

Gehen Sie folgendermaßen vor, um ein Clientzertifikat zu rotieren:

1. Generieren Sie ein neues Client-Zertifikat, indem Sie [clientcertificate:generate](#) der API Gateway REST API oder den Befehl von aufrufen. AWS CLI [generate-client-certificate](#) In diesem Tutorial wird davon ausgegangen, dass die ID des neuen Clientzertifikats laute ndiqef.
2. Aktualisieren Sie den Backend-Server, sodass dieser das neue Clientzertifikat enthält. Entfernen Sie das vorhandene Clientzertifikat noch nicht.

Einige Server erfordern möglicherweise einen Neustart, um die Aktualisierung abzuschließen. Informieren Sie sich in der Serverdokumentation, um festzustellen, ob Sie den Server während der Aktualisierung neu starten müssen.

3. Aktualisieren Sie die API-Stufe, um das neue Client-Zertifikat zu verwenden, indem Sie [stage:update](#) der API Gateway-REST-API mit der neuen Client-Zertifikat-ID (ndiqef) aufrufen:

```
PATCH /restapis/{restapi-id}/stages/stage1 HTTP/1.1
Content-Type: application/json
Host: apigateway.us-east-1.amazonaws.com
X-Amz-Date: 20170603T200400Z
Authorization: AWS4-HMAC-SHA256 Credential=...

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/clientCertificateId",
      "value" : "ndiqef"
    }
  ]
}
```

Sie können auch den [update-stage](#)-Befehl verwenden.

[Wenn Sie eine WebSocket API verwenden, verwenden Sie den Befehl update-stage.apigatewayv2](#)

4. Aktualisieren Sie den Backend-Server, um das alte Zertifikat zu entfernen.
5. Löschen Sie das alte Zertifikat von API Gateway, indem Sie [clientcertificate:delete](#) der API Gateway Gateway-REST-API aufrufen und dabei die clientCertificateId (a1b2c3) des alten Zertifikats angeben:

```
DELETE /clientcertificates/a1b2c3
```

Sie können auch den folgenden Befehl aufrufen: [delete-client-certificate](#)

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Von API Gateway unterstützte Zertifizierungsstellen für HTTP- und HTTP-Proxy-Integrationen

Die folgende Liste zeigt die Autorisierungen, die von API Gateway für HTTP- und HTTP-Proxy-Integrationen unterstützt werden.

Alias name: accvraiz1

SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17

SHA256:

9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1

Alias name: acraizfnmtrcm

SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20

SHA256:

EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F

Alias name: actalis

SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC

SHA256:

55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6

Alias name: actalisauthenticationrootca

SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC

SHA256:

55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6

Alias name: addtrustclass1ca

SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D

SHA256:

8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A

Alias name: addtrustexternalca

SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68

SHA256:

68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F

Alias name: addtrustqualifiedca

SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF

SHA256:

80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1

Alias name: affirmtrustcommercial

SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7

SHA256:

03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A

Alias name: affirmtrustcommercialca

SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7

SHA256:

03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A

Alias name: affirmtrustnetworking

```
SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
SHA256:
0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustnetworkingca
SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
SHA256:
0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustpremium
SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
SHA256:
70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumca
SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
SHA256:
70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumecc
SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
SHA256:
BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: affirmtrustpremiumeccca
SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
SHA256:
BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: amazon-ca-g4-acm1
SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
SHA256:
B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8
Alias name: amazon-ca-g4-acm2
SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
SHA256:
D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B
Alias name: amazon-ca-g4-acm3
SHA1: 7A:DB:56:57:5F:D6:EE:67:85:0A:64:BB:1C:E9:E4:B0:9A:DB:9D:07
SHA256:
6B:EB:9D:20:2E:C2:00:70:BD:D2:5E:D3:C0:C8:33:2C:B4:78:07:C5:82:94:4E:7E:23:28:22:71:A4:8E:0E:C
Alias name: amazon-ca-g4-legacy
SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
SHA256:
CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5
Alias name: amazon-root-ca-ecc-384-1
SHA1: F9:5E:4A:AB:9C:2D:57:61:63:3D:B2:57:B4:0F:24:9E:7B:E2:23:7D
SHA256:
C6:BD:E5:66:C2:72:2A:0E:96:E9:C1:2C:BF:38:92:D9:55:4D:29:03:57:30:72:40:7F:4E:70:17:3B:3C:9B:6
Alias name: amazon-root-ca-rsa-2k-1
```

```
SHA1: 8A:9A:AC:27:FC:86:D4:50:23:AD:D5:63:F9:1E:AE:2C:AF:63:08:6C
SHA256:
0F:8F:33:83:FB:70:02:89:49:24:E1:AA:B0:D7:FB:5A:BF:98:DF:75:8E:0F:FE:61:86:92:BC:F0:75:35:CC:8
Alias name: amazon-root-ca-rsa-4k-1
SHA1: EC:BD:09:61:F5:7A:B6:A8:76:BB:20:8F:14:05:ED:7E:70:ED:39:45
SHA256:
36:AE:AD:C2:6A:60:07:90:6B:83:A3:73:2D:D1:2B:D4:00:5E:C7:F2:76:11:99:A9:D4:DA:63:2F:59:B2:8B:C
Alias name: amazon1
SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
SHA256:
8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazon2
SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
SHA256:
1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazon3
SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
SHA256:
18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazon4
SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
SHA256:
E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amazonrootca1
SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
SHA256:
8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazonrootca2
SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
SHA256:
1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazonrootca3
SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
SHA256:
18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazonrootca4
SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
SHA256:
E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amzninternalinfoseccag3
SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
SHA256:
81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6
Alias name: amzninternalrootca
```

```
SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
SHA256:
0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A
Alias name: aolrootca1
SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
SHA256:
77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: aolrootca2
SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
SHA256:
7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: atostrustedroot2011
SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
SHA256:
F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
SHA256:
04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: baltimorecodesigningca
SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
SHA256:
A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8
Alias name: baltimorecybertrustca
SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
SHA256:
16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: baltimorecybertrustroot
SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
SHA256:
16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: buypassclass2ca
SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
SHA256:
9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass2rootca
SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
SHA256:
9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass3ca
SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
SHA256:
ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: buypassclass3rootca
```

```
SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
SHA256:
ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: cadisigrootr2
SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
SHA256:
E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: camerfirmachambersca
SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
SHA256:
06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: camerfirmachamberscommerceca
SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
SHA256:
0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: camerfirmachamberssignca
SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
SHA256:
13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: certigna
SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
SHA256:
E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: certignarootca
SHA1: 2D:0D:52:14:FF:9E:AD:99:24:01:74:20:47:6E:6C:85:27:27:F5:43
SHA256:
D4:8D:3D:23:EE:DB:50:A4:59:E5:51:97:60:1C:27:77:4B:9D:7B:18:C9:4D:5A:05:95:11:A1:02:50:B9:31:6
Alias name: certplusclass2primaryca
SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
SHA256:
0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: certplusclass3ppprimaryca
SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
SHA256:
CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8
Alias name: certsignrootca
SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
SHA256:
EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: certsignrootcag2
SHA1: 26:F9:93:B4:ED:3D:28:27:B0:B9:4B:A7:E9:15:1D:A3:8D:92:E5:32
SHA256:
65:7C:FE:2F:A7:3F:AA:38:46:25:71:F3:32:A2:36:3A:46:FC:E7:02:09:51:71:07:02:CD:FB:B6:EE:DA:33:0
Alias name: certum2
```

```
SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
SHA256:
B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: certumca
SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
SHA256:
D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: certumtrustednetworkca
SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
SHA256:
5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: certumtrustednetworkca2
SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
SHA256:
B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: cfcaevroot
SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
SHA256:
5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F
Alias name: chambersofcommerceroott2008
SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
SHA256:
06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: chungwaepkirootca
SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
SHA256:
C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: cia-crt-g3-01-ca
SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
SHA256:
20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E
Alias name: cia-crt-g3-02-ca
SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
SHA256:
93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C
Alias name: comodo-ca
SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
SHA256:
52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: comodoaaaca
SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
SHA256:
D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodoaaaservicesroot
```



```
SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
SHA256:
D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodocertificationauthority
SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
SHA256:
0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: comodoecccertificationauthority
SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
SHA256:
17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: comodorsacertificationauthority
SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
SHA256:
52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: cybertrustglobalroot
SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
SHA256:
96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: deprecateditsecca
SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
SHA256:
9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C
Alias name: deutschetelekomrootca2
SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
SHA256:
B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: digicertassuredidrootca
SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
SHA256:
3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: digicertassuredidrootg2
SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
SHA256:
7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8
Alias name: digicertassuredidrootg3
SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
SHA256:
7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C
Alias name: digicertglobalrootca
SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
SHA256:
43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: digicertglobalrootg2
```

```
SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
SHA256:
CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5
Alias name: digicertglobalrootg3
SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
SHA256:
31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D
Alias name: digicerthighassuranceevrootca
SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
SHA256:
74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: digicerttrustedrootg4
SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
SHA256:
55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8
Alias name: dstrootcac3
SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
SHA256:
06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: dtrustrootclass3ca22009
SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
SHA256:
49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: dtrustrootclass3ca2ev2009
SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
SHA256:
EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: ecacc
SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
SHA256:
88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9
Alias name: emsigneccrootcac3
SHA1: B6:AF:43:C2:9B:81:53:7D:F6:EF:6B:C3:1F:1F:60:15:0C:EE:48:66
SHA256:
BC:4D:80:9B:15:18:9D:78:DB:3E:1D:8C:F4:F9:72:6A:79:5D:A1:64:3C:A5:F1:35:8E:1D:DB:0E:DC:0D:7E:B
Alias name: emsigneccrootcag3
SHA1: 30:43:FA:4F:F2:57:DC:A0:C3:80:EE:2E:58:EA:78:B2:3F:E6:BB:C1
SHA256:
86:A1:EC:BA:08:9C:4A:8D:3B:BE:27:34:C6:12:BA:34:1D:81:3E:04:3C:F9:E8:A8:62:CD:5C:57:A3:6B:BE:6
Alias name: emsignrootcac1
SHA1: E7:2E:F1:DF:FC:B2:09:28:CF:5D:D4:D5:67:37:B1:51:CB:86:4F:01
SHA256:
12:56:09:AA:30:1D:A0:A2:49:B9:7A:82:39:CB:6A:34:21:6F:44:DC:AC:9F:39:54:B1:42:92:F2:E8:C8:60:8
Alias name: emsignrootcag1
```

```
SHA1: 8A:C7:AD:8F:73:AC:4E:C1:B5:75:4D:A5:40:F4:FC:CF:7C:B5:8E:8C
SHA256:
40:F6:AF:03:46:A9:9A:A1:CD:1D:55:5A:4E:9C:CE:62:C7:F9:63:46:03:EE:40:66:15:83:3D:C8:C8:D0:03:6
Alias name: entrust2048ca
SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
SHA256:
6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustevca
SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
SHA256:
73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustnetpremium2048secureserverca
SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
SHA256:
6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustrootcag2
SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
SHA256:
43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthority
SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
SHA256:
73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustrootcertificationauthorityec1
SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
SHA256:
02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F
Alias name: entrustrootcertificationauthorityg2
SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
SHA256:
43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthorityg4
SHA1: 14:88:4E:86:26:37:B0:26:AF:59:62:5C:40:77:EC:35:29:BA:96:01
SHA256:
DB:35:17:D1:F6:73:2A:2D:5A:B9:7C:53:3E:C7:07:79:EE:32:70:A6:2F:B4:AC:42:38:37:24:60:E6:F0:1E:8
Alias name: epkirootcertificationauthority
SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
SHA256:
C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: equifaxsecureebusinessca1
SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
SHA256:
2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9
Alias name: equifaxsecureglobalebusinessca1
```

```
SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
SHA256:
86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A
Alias name: eszignorootca2017
SHA1: 89:D4:83:03:4F:9E:9A:48:80:5F:72:37:D4:A9:A6:EF:CB:7C:1F:D1
SHA256:
BE:B0:0B:30:83:9B:9B:C3:2C:32:E4:44:79:05:95:06:41:F2:64:21:B1:5E:D0:89:19:8B:51:8A:E2:EA:1B:9
Alias name: etugracertificationauthority
SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
SHA256:
B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: gd-class2-root.pem
SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
SHA256:
C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: gd_bundle-g2.pem
SHA1: 27:AC:93:69:FA:F2:52:07:BB:26:27:CE:FA:CC:BE:4E:F9:C3:19:B8
SHA256:
97:3A:41:27:6F:FD:01:E0:27:A2:AA:D4:9E:34:C3:78:46:D3:E9:76:FF:6A:62:0B:67:12:E3:38:32:04:1A:A
Alias name: gdcatrustauthr5root
SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
SHA256:
BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9
Alias name: gdroot-g2.pem
SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
SHA256:
45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: geotrustglobalca
SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
SHA256:
FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: geotrustprimaryca
SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
SHA256:
37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycag2
SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
SHA256:
5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycag3
SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
SHA256:
B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustprimarycertificationauthority
```

```
SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
SHA256:
37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycertificationauthorityg2
SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
SHA256:
5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycertificationauthorityg3
SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
SHA256:
B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustuniversalca
SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
SHA256:
A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: geotrustuniversalca2
SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
SHA256:
A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: globalchambersignroot2008
SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
SHA256:
13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: globalsignca
SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
SHA256:
EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsigneccrootcar4
SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
SHA256:
BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8
Alias name: globalsigneccrootcar5
SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
SHA256:
17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2
Alias name: globalsignr2ca
SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
SHA256:
CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignr3ca
SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
SHA256:
CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootca
```

```
SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
SHA256:
EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsignrootcar2
SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
SHA256:
CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignrootcar3
SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
SHA256:
CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootcar6
SHA1: 80:94:64:0E:B5:A7:A1:CA:11:9C:1F:DD:D5:9F:81:02:63:A7:FB:D1
SHA256:
2C:AB:EA:FE:37:D0:6C:A2:2A:BA:73:91:C0:03:3D:25:98:29:52:C4:53:64:73:49:76:3A:3A:B5:AD:6C:CF:6
Alias name: godaddyclass2ca
SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
SHA256:
C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: godaddyrootcertificateauthorityg2
SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
SHA256:
45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: godaddyrootg2ca
SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
SHA256:
45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: gtsrootr1
SHA1: E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
SHA256:
2A:57:54:71:E3:13:40:BC:21:58:1C:BD:2C:F1:3E:15:84:63:20:3E:CE:94:BC:F9:D3:CC:19:6B:F0:9A:54:7
Alias name: gtsrootr2
SHA1: D2:73:96:2A:2A:5E:39:9F:73:3F:E1:C7:1E:64:3F:03:38:34:FC:4D
SHA256:
C4:5D:7B:B0:8E:6D:67:E6:2E:42:35:11:0B:56:4E:5F:78:FD:92:EF:05:8C:84:0A:EA:4E:64:55:D7:58:5C:6
Alias name: gtsrootr3
SHA1: 30:D4:24:6F:07:FF:DB:91:89:8A:0B:E9:49:66:11:EB:8C:5E:46:E5
SHA256:
15:D5:B8:77:46:19:EA:7D:54:CE:1C:A6:D0:B0:C4:03:E0:37:A9:17:F1:31:E8:A0:4E:1E:6B:7A:71:BA:BC:E
Alias name: gtsrootr4
SHA1: 2A:1D:60:27:D9:4A:B1:0A:1C:4D:91:5C:CD:33:A0:CB:3E:2D:54:CB
SHA256:
71:CC:A5:39:1F:9E:79:4B:04:80:25:30:B3:63:E1:21:DA:8A:30:43:BB:26:66:2F:EA:4D:CA:7F:C9:51:A4:B
Alias name: hellenicacademicandresearchinstitutionseccrootca2015
```

```
SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
SHA256:
44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3
Alias name: hellenicacademicandresearchinstitutionsrootca2011
SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
SHA256:
BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: hellenicacademicandresearchinstitutionsrootca2015
SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
SHA256:
A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
Alias name: hongkongpostrootca1
SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
SHA256:
F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: hongkongpostrootca3
SHA1: 58:A2:D0:EC:20:52:81:5B:C1:F3:F8:64:02:24:4E:C2:8E:02:4B:02
SHA256:
5A:2F:C0:3F:0C:83:B0:90:BB:FA:40:60:4B:09:88:44:6C:76:36:18:3D:F9:84:6E:17:10:1A:44:7F:B8:EF:D
Alias name: identrustcommercialrootca1
SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
SHA256:
5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A
Alias name: identrustpublicsectorrootca1
SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
SHA256:
30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2
Alias name: isrgrootx1
SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
SHA256:
96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C
Alias name: izenpecom
SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
SHA256:
25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: keynectisrootca
SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
SHA256:
42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3
Alias name: microseceszignorrootca2009
SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
SHA256:
3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert0.pem
```

```
SHA1: 97:81:79:50:D8:1C:96:70:CC:34:D8:09:CF:79:44:31:36:7E:F4:74
SHA256:
A5:31:25:18:8D:21:10:AA:96:4B:02:C7:B7:C6:DA:32:03:17:08:94:E5:FB:71:FF:FB:66:67:D5:E6:81:0A:3
Alias name: mozillacert1.pem
SHA1: 23:E5:94:94:51:95:F2:41:48:03:B4:D5:64:D2:A3:A3:F5:D8:8B:8C
SHA256:
B4:41:0B:73:E2:E6:EA:CA:47:FB:C4:2F:8F:A4:01:8A:F4:38:1D:C5:4C:FA:A8:44:50:46:1E:ED:09:45:4D:E
Alias name: mozillacert10.pem
SHA1: 5F:3A:FC:0A:8B:64:F6:86:67:34:74:DF:7E:A9:A2:FE:F9:FA:7A:51
SHA256:
21:DB:20:12:36:60:BB:2E:D4:18:20:5D:A1:1E:E7:A8:5A:65:E2:BC:6E:55:B5:AF:7E:78:99:C8:A2:66:D9:2
Alias name: mozillacert100.pem
SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
SHA256:
49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: mozillacert101.pem
SHA1: 99:A6:9B:E6:1A:FE:88:6B:4D:2B:82:00:7C:B8:54:FC:31:7E:15:39
SHA256:
62:F2:40:27:8C:56:4C:4D:D8:BF:7D:9D:4F:6F:36:6E:A8:94:D2:2F:5F:34:D9:89:A9:83:AC:EC:2F:FF:ED:5
Alias name: mozillacert102.pem
SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
SHA256:
EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: mozillacert103.pem
SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
SHA256:
3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B
Alias name: mozillacert104.pem
SHA1: 4F:99:AA:93:FB:2B:D1:37:26:A1:99:4A:CE:7F:F0:05:F2:93:5D:1E
SHA256:
1C:01:C6:F4:DB:B2:FE:FC:22:55:8B:2B:CA:32:56:3F:49:84:4A:CF:C3:2B:7B:E4:B0:FF:59:9F:9E:8C:7A:F
Alias name: mozillacert105.pem
SHA1: 77:47:4F:C6:30:E4:0F:4C:47:64:3F:84:BA:B8:C6:95:4A:8A:41:EC
SHA256:
F0:9B:12:2C:71:14:F4:A0:9B:D4:EA:4F:4A:99:D5:58:B4:6E:4C:25:CD:81:14:0D:29:C0:56:13:91:4C:38:4
Alias name: mozillacert106.pem
SHA1: E7:A1:90:29:D3:D5:52:DC:0D:0F:C6:92:D3:EA:88:0D:15:2E:1A:6B
SHA256:
D9:5F:EA:3C:A4:EE:DC:E7:4C:D7:6E:75:FC:6D:1F:F6:2C:44:1F:0F:A8:BC:77:F0:34:B1:9E:5D:B2:58:01:5
Alias name: mozillacert107.pem
SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
SHA256:
F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C
Alias name: mozillacert108.pem
```



```
SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
SHA256:
EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: mozillacert109.pem
SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
SHA256:
E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: mozillacert11.pem
SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
SHA256:
3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: mozillacert110.pem
SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
SHA256:
9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: mozillacert111.pem
SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
SHA256:
59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: mozillacert112.pem
SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
SHA256:
DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: mozillacert113.pem
SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
SHA256:
6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: mozillacert114.pem
SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
SHA256:
B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: mozillacert115.pem
SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
SHA256:
91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: mozillacert116.pem
SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
SHA256:
F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: mozillacert117.pem
SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
SHA256:
16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: mozillacert118.pem
```

```
SHA1: 7E:78:4A:10:1C:82:65:CC:2D:E1:F1:6D:47:B4:40:CA:D9:0A:19:45
SHA256:
5F:0B:62:EA:B5:E3:53:EA:65:21:65:16:58:FB:B6:53:59:F4:43:28:0A:4A:FB:D1:04:D7:7D:10:F9:F0:4C:0
Alias name: mozillacert119.pem
SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
SHA256:
CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: mozillacert12.pem
SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
SHA256:
43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: mozillacert120.pem
SHA1: DA:40:18:8B:91:89:A3:ED:EE:AE:DA:97:FE:2F:9D:F5:B7:D1:8A:41
SHA256:
CF:56:FF:46:A4:A1:86:10:9D:D9:65:84:B5:EE:B5:8A:51:0C:42:75:B0:E5:F9:4F:40:BB:AE:86:5E:19:F6:7
Alias name: mozillacert121.pem
SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
SHA256:
8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: mozillacert122.pem
SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
SHA256:
68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: mozillacert123.pem
SHA1: 2A:B6:28:48:5E:78:FB:F3:AD:9E:79:10:DD:6B:DF:99:72:2C:96:E5
SHA256:
07:91:CA:07:49:B2:07:82:AA:D3:C7:D7:BD:0C:DF:C9:48:58:35:84:3E:B2:D7:99:60:09:CE:43:AB:6C:69:2
Alias name: mozillacert124.pem
SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
SHA256:
80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: mozillacert125.pem
SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
SHA256:
73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: mozillacert126.pem
SHA1: 25:01:90:19:CF:FB:D9:99:1C:B7:68:25:74:8D:94:5F:30:93:95:42
SHA256:
AF:8B:67:62:A1:E5:28:22:81:61:A9:5D:5C:55:9E:E2:66:27:8F:75:D7:9E:83:01:89:A5:03:50:6A:BD:6B:4
Alias name: mozillacert127.pem
SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
SHA256:
FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: mozillacert128.pem
```

```
SHA1: A9:E9:78:08:14:37:58:88:F2:05:19:B0:6D:2B:0D:2B:60:16:90:7D
SHA256:
CA:2D:82:A0:86:77:07:2F:8A:B6:76:4F:F0:35:67:6C:FE:3E:5E:32:5E:01:21:72:DF:3F:92:09:6D:B7:9B:8
Alias name: mozillacert129.pem
SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
SHA256:
A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: mozillacert13.pem
SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
SHA256:
6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: mozillacert130.pem
SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
SHA256:
F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: mozillacert131.pem
SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
SHA256:
A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: mozillacert132.pem
SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
SHA256:
77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: mozillacert133.pem
SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
SHA256:
7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: mozillacert134.pem
SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
SHA256:
69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2
Alias name: mozillacert135.pem
SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
SHA256:
D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: mozillacert136.pem
SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
SHA256:
D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: mozillacert137.pem
SHA1: 4A:65:D5:F4:1D:EF:39:B8:B8:90:4A:4A:D3:64:81:33:CF:C7:A1:D1
SHA256:
BD:81:CE:3B:4F:65:91:D1:1A:67:B5:FC:7A:47:FD:EF:25:52:1B:F9:AA:4E:18:B9:E3:DF:2E:34:A7:80:3B:E
Alias name: mozillacert138.pem
```

```
SHA1: E1:9F:E3:0E:8B:84:60:9E:80:9B:17:0D:72:A8:C5:BA:6E:14:09:BD
SHA256:
3F:06:E5:56:81:D4:96:F5:BE:16:9E:B5:38:9F:9F:2B:8F:F6:1E:17:08:DF:68:81:72:48:49:CD:5D:27:CB:6
Alias name: mozillacert139.pem
SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
SHA256:
A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: mozillacert14.pem
SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
SHA256:
74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: mozillacert140.pem
SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
SHA256:
85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: mozillacert141.pem
SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
SHA256:
58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: mozillacert142.pem
SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
SHA256:
18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: mozillacert143.pem
SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
SHA256:
E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: mozillacert144.pem
SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
SHA256:
79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: mozillacert145.pem
SHA1: 10:1D:FA:3F:D5:0B:CB:BB:9B:B5:60:0C:19:55:A4:1A:F4:73:3A:04
SHA256:
D4:1D:82:9E:8C:16:59:82:2A:F9:3F:CE:62:BF:FC:DE:26:4F:C8:4E:8B:95:0C:5F:F2:75:D0:52:35:46:95:A
Alias name: mozillacert146.pem
SHA1: 21:FC:BD:8E:7F:6C:AF:05:1B:D1:B3:43:EC:A8:E7:61:47:F2:0F:8A
SHA256:
48:98:C6:88:8C:0C:FF:B0:D3:E3:1A:CA:8A:37:D4:E3:51:5F:F7:46:D0:26:35:D8:66:46:CF:A0:A3:18:5A:E
Alias name: mozillacert147.pem
SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
SHA256:
85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: mozillacert148.pem
```

```
SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
SHA256:
6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: mozillacert149.pem
SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
SHA256:
0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: mozillacert15.pem
SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
SHA256:
0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: mozillacert150.pem
SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
SHA256:
EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E
Alias name: mozillacert151.pem
SHA1: AC:ED:5F:65:53:FD:25:CE:01:5F:1F:7A:48:3B:6A:74:9F:61:78:C6
SHA256:
7F:12:CD:5F:7E:5E:29:0E:C7:D8:51:79:D5:B7:2C:20:A5:BE:75:08:FF:DB:5B:F8:1A:B9:68:4A:7F:C9:F6:6
Alias name: mozillacert16.pem
SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
SHA256:
06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: mozillacert17.pem
SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
SHA256:
76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4
Alias name: mozillacert18.pem
SHA1: 79:98:A3:08:E1:4D:65:85:E6:C2:1E:15:3A:71:9F:BA:5A:D3:4A:D9
SHA256:
44:04:E3:3B:5E:14:0D:CF:99:80:51:FD:FC:80:28:C7:C8:16:15:C5:EE:73:7B:11:1B:58:82:33:A9:B5:35:A
Alias name: mozillacert19.pem
SHA1: B4:35:D4:E1:11:9D:1C:66:90:A7:49:EB:B3:94:BD:63:7B:A7:82:B7
SHA256:
C4:70:CF:54:7E:23:02:B9:77:FB:29:DD:71:A8:9A:7B:6C:1F:60:77:7B:03:29:F5:60:17:F3:28:BF:4F:6B:E
Alias name: mozillacert2.pem
SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
SHA256:
69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: mozillacert20.pem
SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
SHA256:
62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: mozillacert21.pem
```

```
SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
SHA256:
BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: mozillacert22.pem
SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
SHA256:
37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: mozillacert23.pem
SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
SHA256:
8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: mozillacert24.pem
SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
SHA256:
66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6
Alias name: mozillacert25.pem
SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
SHA256:
9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: mozillacert26.pem
SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
SHA256:
F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: mozillacert27.pem
SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
SHA256:
42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: mozillacert28.pem
SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
SHA256:
0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: mozillacert29.pem
SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
SHA256:
15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: mozillacert30.pem
SHA1: 87:9F:4B:EE:05:DF:98:58:3B:E3:60:D6:33:E7:0D:3F:FE:98:71:AF
SHA256:
39:DF:7B:68:2B:7B:93:8F:84:71:54:81:CC:DE:8D:60:D8:F2:2E:C5:98:87:7D:0A:AA:C1:2B:59:18:2B:03:1
Alias name: mozillacert31.pem
SHA1: E7:B4:F6:9D:61:EC:90:69:DB:7E:90:A7:40:1A:3C:F4:7D:4F:E8:EE
SHA256:
A7:12:72:AE:AA:A3:CF:E8:72:7F:7F:B3:9F:0F:B3:D1:E5:42:6E:90:60:B0:6E:E6:F1:3E:9A:3C:58:33:CD:4
```

```
SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
SHA256:
17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: mozillacert32.pem
SHA1: 60:D6:89:74:B5:C2:65:9E:8A:0F:C1:88:7C:88:D2:46:69:1B:18:2C
SHA256:
B9:BE:A7:86:0A:96:2E:A3:61:1D:AB:97:AB:6D:A3:E2:1C:10:68:B9:7D:55:57:5E:D0:E1:12:79:C1:1C:89:3
Alias name: mozillacert33.pem
SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
SHA256:
A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: mozillacert34.pem
SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
SHA256:
41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: mozillacert35.pem
SHA1: 2A:C8:D5:8B:57:CE:BF:2F:49:AF:F2:FC:76:8F:51:14:62:90:7A:41
SHA256:
92:BF:51:19:AB:EC:CA:D0:B1:33:2D:C4:E1:D0:5F:BA:75:B5:67:90:44:EE:0C:A2:6E:93:1F:74:4F:2F:33:C
Alias name: mozillacert36.pem
SHA1: 23:88:C9:D3:71:CC:9E:96:3D:FF:7D:3C:A7:CE:FC:D6:25:EC:19:0D
SHA256:
32:7A:3D:76:1A:BA:DE:A0:34:EB:99:84:06:27:5C:B1:A4:77:6E:FD:AE:2F:DF:6D:01:68:EA:1C:4F:55:67:D
Alias name: mozillacert37.pem
SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
SHA256:
E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: mozillacert38.pem
SHA1: CB:A1:C5:F8:B0:E3:5E:B8:B9:45:12:D3:F9:34:A2:E9:06:10:D3:36
SHA256:
A6:C5:1E:0D:A5:CA:0A:93:09:D2:E4:C0:E4:0C:2A:F9:10:7A:AE:82:03:85:7F:E1:98:E3:E7:69:E3:43:08:5
Alias name: mozillacert39.pem
SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
SHA256:
E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: mozillacert40.pem
SHA1: E3:92:51:2F:0A:CF:F5:05:DF:F6:DE:06:7F:75:37:E1:65:EA:57:4B
SHA256:
0B:5E:ED:4E:84:64:03:CF:55:E0:65:84:84:40:ED:2A:82:75:8B:F5:B9:AA:1F:25:3D:46:13:CF:A0:80:FF:3
Alias name: mozillacert41.pem
```

```
SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
SHA256:
EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: mozillacert42.pem
SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
SHA256:
B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: mozillacert43.pem
SHA1: F9:CD:0E:2C:DA:76:24:C1:8F:BD:F0:F0:AB:B6:45:B8:F7:FE:D5:7A
SHA256:
50:79:41:C7:44:60:A0:B4:70:86:22:0D:4E:99:32:57:2A:B5:D1:B5:BB:CB:89:80:AB:1C:B1:76:51:A8:44:D
Alias name: mozillacert44.pem
SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
SHA256:
96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: mozillacert45.pem
SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
SHA256:
C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: mozillacert46.pem
SHA1: 40:9D:4B:D9:17:B5:5C:27:B6:9B:64:CB:98:22:44:0D:CD:09:B8:89
SHA256:
EC:C3:E9:C3:40:75:03:BE:E0:91:AA:95:2F:41:34:8F:F8:8B:AA:86:3B:22:64:BE:FA:C8:07:90:15:74:E9:3
Alias name: mozillacert47.pem
SHA1: 1B:4B:39:61:26:27:6B:64:91:A2:68:6D:D7:02:43:21:2D:1F:1D:96
SHA256:
E4:C7:34:30:D7:A5:B5:09:25:DF:43:37:0A:0D:21:6E:9A:79:B9:D6:DB:83:73:A0:C6:9E:B1:CC:31:C7:C5:2
Alias name: mozillacert48.pem
SHA1: A0:A1:AB:90:C9:FC:84:7B:3B:12:61:E8:97:7D:5F:D3:22:61:D3:CC
SHA256:
0F:4E:9C:DD:26:4B:02:55:50:D1:70:80:63:40:21:4F:E9:44:34:C9:B0:2F:69:7E:C7:10:FC:5F:EA:FB:5E:3
Alias name: mozillacert49.pem
SHA1: 61:57:3A:11:DF:0E:D8:7E:D5:92:65:22:EA:D0:56:D7:44:B3:23:71
SHA256:
B7:B1:2B:17:1F:82:1D:AA:99:0C:D0:FE:50:87:B1:28:44:8B:A8:E5:18:4F:84:C5:1E:02:B5:C8:FB:96:2B:2
Alias name: mozillacert50.pem
SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
SHA256:
CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: mozillacert51.pem
SHA1: 8C:96:BA:EB:DD:2B:07:07:48:EE:30:32:66:A0:F3:98:6E:7C:AE:58
SHA256:
35:AE:5B:DD:D8:F7:AE:63:5C:FF:BA:56:82:A8:F0:0B:95:F4:84:62:C7:10:8E:E9:A0:E5:29:2B:07:4A:AF:B
```



```
SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
SHA256:
EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: mozillacert52.pem
SHA1: 8B:AF:4C:9B:1D:F0:2A:92:F7:DA:12:8E:B9:1B:AC:F4:98:60:4B:6F
SHA256:
E2:83:93:77:3D:A8:45:A6:79:F2:08:0C:C7:FB:44:A3:B7:A1:C3:79:2C:B7:EB:77:29:FD:CB:6A:8D:99:AE:A
Alias name: mozillacert53.pem
SHA1: 7F:8A:B0:CF:D0:51:87:6A:66:F3:36:0F:47:C8:8D:8C:D3:35:FC:74
SHA256:
2D:47:43:7D:E1:79:51:21:5A:12:F3:C5:8E:51:C7:29:A5:80:26:EF:1F:CC:0A:5F:B3:D9:DC:01:2F:60:0D:1
Alias name: mozillacert54.pem
SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
SHA256:
B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: mozillacert55.pem
SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
SHA256:
A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: mozillacert56.pem
SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
SHA256:
4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: mozillacert57.pem
SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
SHA256:
F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: mozillacert58.pem
SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
SHA256:
5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: mozillacert59.pem
SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
SHA256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: mozillacert60.pem
SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
SHA256:
C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: mozillacert61.pem
SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
SHA256:
BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: mozillacert61.pem
```

```
SHA1: E0:B4:32:2E:B2:F6:A5:68:B6:54:53:84:48:18:4A:50:36:87:43:84
SHA256:
03:95:0F:B4:9A:53:1F:3E:19:91:94:23:98:DF:A9:E0:EA:32:D7:BA:1C:DD:9B:C8:5D:B5:7E:D9:40:0B:43:4
Alias name: mozillacert62.pem
SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
SHA256:
A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: mozillacert63.pem
SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
SHA256:
3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert64.pem
SHA1: 62:7F:8D:78:27:65:63:99:D2:7D:7F:90:44:C9:FE:B3:F3:3E:FA:9A
SHA256:
AB:70:36:36:5C:71:54:AA:29:C2:C2:9F:5D:41:91:16:3B:16:2A:22:25:01:13:57:D5:6D:07:FF:A7:BC:1F:7
Alias name: mozillacert65.pem
SHA1: 69:BD:8C:F4:9C:D3:00:FB:59:2E:17:93:CA:55:6A:F3:EC:AA:35:FB
SHA256:
BC:23:F9:8A:31:3C:B9:2D:E3:BB:FC:3A:5A:9F:44:61:AC:39:49:4C:4A:E1:5A:9E:9D:F1:31:E9:9B:73:01:9
Alias name: mozillacert66.pem
SHA1: DD:E1:D2:A9:01:80:2E:1D:87:5E:84:B3:80:7E:4B:B1:FD:99:41:34
SHA256:
E6:09:07:84:65:A4:19:78:0C:B6:AC:4C:1C:0B:FB:46:53:D9:D9:CC:6E:B3:94:6E:B7:F3:D6:99:97:BA:D5:9
Alias name: mozillacert67.pem
SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
SHA256:
CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: mozillacert68.pem
SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
SHA256:
04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: mozillacert69.pem
SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
SHA256:
25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: mozillacert70.pem
SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
SHA256:
14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: mozillacert71.pem
```

```
SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
SHA256:
13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: mozillacert72.pem
SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
SHA256:
45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: mozillacert73.pem
SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
SHA256:
2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: mozillacert74.pem
SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
SHA256:
56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: mozillacert75.pem
SHA1: D2:32:09:AD:23:D3:14:23:21:74:E4:0D:7F:9D:62:13:97:86:63:3A
SHA256:
08:29:7A:40:47:DB:A2:36:80:C7:31:DB:6E:31:76:53:CA:78:48:E1:BE:BD:3A:0B:01:79:A7:07:F9:2C:F1:7
Alias name: mozillacert76.pem
SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
SHA256:
03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: mozillacert77.pem
SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
SHA256:
EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: mozillacert78.pem
SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
SHA256:
0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: mozillacert79.pem
SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
SHA256:
70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: mozillacert80.pem
SHA1: 3E:2B:F7:F2:03:1B:96:F3:8C:E6:C4:D8:A8:5D:3E:2D:58:47:6A:0F
SHA256:
C7:66:A9:BE:F2:D4:07:1C:86:3A:31:AA:49:20:E8:13:B2:D1:98:60:8C:B7:B7:CF:E2:11:43:B8:36:DF:09:E
Alias name: mozillacert81.pem
```

```
SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
SHA256:
5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: mozillacert82.pem
SHA1: 2E:14:DA:EC:28:F0:FA:1E:8E:38:9A:4E:AB:EB:26:C0:0A:D3:83:C3
SHA256:
FC:BF:E2:88:62:06:F7:2B:27:59:3C:8B:07:02:97:E1:2D:76:9E:D1:0E:D7:93:07:05:A8:09:8E:FF:C1:4D:1
Alias name: mozillacert83.pem
SHA1: A0:73:E5:C5:BD:43:61:0D:86:4C:21:13:0A:85:58:57:CC:9C:EA:46
SHA256:
8C:4E:DF:D0:43:48:F3:22:96:9E:7E:29:A4:CD:4D:CA:00:46:55:06:1C:16:E1:B0:76:42:2E:F3:42:AD:63:0
Alias name: mozillacert84.pem
SHA1: D3:C0:63:F2:19:ED:07:3E:34:AD:5D:75:0B:32:76:29:FF:D5:9A:F2
SHA256:
79:3C:BF:45:59:B9:FD:E3:8A:B2:2D:F1:68:69:F6:98:81:AE:14:C4:B0:13:9A:C7:88:A7:8A:1A:FC:CA:02:F
Alias name: mozillacert85.pem
SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
SHA256:
BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: mozillacert86.pem
SHA1: 74:2C:31:92:E6:07:E4:24:EB:45:49:54:2B:E1:BB:C5:3E:61:74:E2
SHA256:
E7:68:56:34:EF:AC:F6:9A:CE:93:9A:6B:25:5B:7B:4F:AB:EF:42:93:5B:50:A2:65:AC:B5:CB:60:27:E4:4E:7
Alias name: mozillacert87.pem
SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
SHA256:
51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: mozillacert88.pem
SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
SHA256:
BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: mozillacert89.pem
SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
SHA256:
E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: mozillacert90.pem
SHA1: F4:8B:11:BF:DE:AB:BE:94:54:20:71:E6:41:DE:6B:BE:88:2B:40:B9
SHA256:
76:00:29:5E:EF:E8:5B:9E:1F:D6:24:DB:76:06:2A:AA:AE:59:81:8A:54:D2:77:4C:D4:C0:B2:C0:11:31:E1:B
Alias name: mozillacert91.pem
```

```
SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
SHA256:
C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: mozillacert92.pem
SHA1: A3:F1:33:3F:E2:42:BF:CF:C5:D1:4E:8F:39:42:98:40:68:10:D1:A0
SHA256:
E1:78:90:EE:09:A3:FB:F4:F4:8B:9C:41:4A:17:D6:37:B7:A5:06:47:E9:BC:75:23:22:72:7F:CC:17:42:A9:1
Alias name: mozillacert93.pem
SHA1: 31:F1:FD:68:22:63:20:EE:C6:3B:3F:9D:EA:4A:3E:53:7C:7C:39:17
SHA256:
C7:BA:65:67:DE:93:A7:98:AE:1F:AA:79:1E:71:2D:37:8F:AE:1F:93:C4:39:7F:EA:44:1B:B7:CB:E6:FD:59:9
Alias name: mozillacert94.pem
SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
SHA256:
9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: mozillacert95.pem
SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
SHA256:
ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: mozillacert96.pem
SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
SHA256:
FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: mozillacert97.pem
SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
SHA256:
83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: mozillacert98.pem
SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
SHA256:
3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7
Alias name: mozillacert99.pem
SHA1: F1:7F:6F:B6:31:DC:99:E3:A3:C8:7F:FE:1C:F1:81:10:88:D9:60:33
SHA256:
97:8C:D9:66:F2:FA:A0:7B:A7:AA:95:00:D9:C0:2E:9D:77:F2:CD:AD:A6:AD:6B:A7:4A:F4:B9:1C:66:59:3C:5
Alias name: netlockaranyclassgoldfotanusitvany
SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
SHA256:
6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: networksolutionscertificateauthority
SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
SHA256:
15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: oistewisekeyglobalrootgaca
```

```
SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
SHA256:
41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: oistewisekeyglobalrootgbca
SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
SHA256:
6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D
Alias name: oistewisekeyglobalrootgcca
SHA1: E0:11:84:5E:34:DE:BE:88:81:B9:9C:F6:16:26:D1:96:1F:C3:B9:31
SHA256:
85:60:F9:1C:36:24:DA:BA:95:70:B5:FE:A0:DB:E3:6F:F1:1A:83:23:BE:94:86:85:4F:B3:F3:4A:55:71:19:8
Alias name: quovadisrootca
SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
SHA256:
A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: quovadisrootca1g3
SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
SHA256:
8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7
Alias name: quovadisrootca2
SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
SHA256:
85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: quovadisrootca2g3
SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
SHA256:
8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4
Alias name: quovadisrootca3
SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
SHA256:
18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: quovadisrootca3g3
SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
SHA256:
88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4
Alias name: secomevrootca1
SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
SHA256:
A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: secomscrootca1
SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
SHA256:
E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: secomscrootca2
```

```
SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
SHA256:
51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: secomvalicertclass1ca
SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
SHA256:
F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: secureglobalca
SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
SHA256:
42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: securesignrootca11
SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
SHA256:
BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: securetrustca
SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
SHA256:
F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: securitycommunicationrootca
SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
SHA256:
E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: securitycommunicationrootca2
SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
SHA256:
51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: soneraclass1ca
SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
SHA256:
CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3
Alias name: soneraclass2ca
SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
SHA256:
79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: soneraclass2rootca
SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
SHA256:
79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: sslcomevrootcertificationauthorityecc
SHA1: 4C:DD:51:A3:D1:F5:20:32:14:B0:C6:C5:32:23:03:91:C7:46:42:6D
SHA256:
22:A2:C1:F7:BD:ED:70:4C:C1:E7:01:B5:F4:08:C3:10:88:0F:E9:56:B5:DE:2A:4A:44:F9:9C:87:3A:25:A7:C
Alias name: sslcomevrootcertificationauthorityrsar2
```

```
SHA1: 74:3A:F0:52:9B:D0:32:A0:F4:4A:83:CD:D4:BA:A9:7B:7C:2E:C4:9A
SHA256:
2E:7B:F1:6C:C2:24:85:A7:BB:E2:AA:86:96:75:07:61:B0:AE:39:BE:3B:2F:E9:D0:CC:6D:4E:F7:34:91:42:5
Alias name: sslcomrootcertificationauthorityecc
SHA1: C3:19:7C:39:24:E6:54:AF:1B:C4:AB:20:95:7A:E2:C3:0E:13:02:6A
SHA256:
34:17:BB:06:CC:60:07:DA:1B:96:1C:92:0B:8A:B4:CE:3F:AD:82:0E:4A:A3:0B:9A:CB:C4:A7:4E:BD:CE:BC:6
Alias name: sslcomrootcertificationauthorityrsa
SHA1: B7:AB:33:08:D1:EA:44:77:BA:14:80:12:5A:6F:BD:A9:36:49:0C:BB
SHA256:
85:66:6A:56:2E:E0:BE:5C:E9:25:C1:D8:89:0A:6F:76:A8:7E:C1:6D:4D:7D:5F:29:EA:74:19:CF:20:12:3B:6
Alias name: staatdernederlandenevrootca
SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
SHA256:
4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5
Alias name: staatdernederlandenrootcag3
SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
SHA256:
3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2
Alias name: starfieldclass2ca
SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
SHA256:
14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: starfieldrootcertificateauthorityg2
SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
SHA256:
2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldrootg2ca
SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
SHA256:
2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldservicesrootcertificateauthorityg2
SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
SHA256:
56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: starfieldservicesrootg2ca
SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
SHA256:
56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: swisssigngoldcag2
SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
SHA256:
62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssigngoldg2ca
```



```
SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
SHA256:
62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssignplatinumg2ca
SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
SHA256:
3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3
Alias name: swisssignsilvercag2
SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
SHA256:
BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: swisssignsilverg2ca
SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
SHA256:
BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: szafirrootca2
SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
SHA256:
A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F
Alias name: teliasonerarootcav1
SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
SHA256:
DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: thawtepersonalfreemailca
SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
SHA256:
5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8
Alias name: thawtepremiumserverca
SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
SHA256:
3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E
Alias name: thawteprimaryrootca
SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
SHA256:
8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: thawteprimaryrootcag2
SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
SHA256:
A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: thawteprimaryrootcag3
SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
SHA256:
4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: thawteserverca
```

```
SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
SHA256:
87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6
Alias name: trustcenterclass2caii
SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
SHA256:
E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: trustcenterclass4caii
SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
SHA256:
32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3
Alias name: trustcenteruniversalcai
SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
SHA256:
EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: trustcoreca1
SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
SHA256:
5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9
Alias name: trustcorrootcertca1
SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
SHA256:
D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5
Alias name: trustcorrootcertca2
SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
SHA256:
07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6
Alias name: trustisfypsrootca
SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
SHA256:
C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: ttelesecglobalrootclass2
SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
SHA256:
91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass2ca
SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
SHA256:
91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass3
SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
SHA256:
FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: ttelesecglobalrootclass3ca
```

```
SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
SHA256:
FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: tubitakkamusmssllkoksertifikasisurum1
SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
SHA256:
46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1
Alias name: twcaglobalrootca
SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
SHA256:
59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: twcarootcertificationauthority
SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
SHA256:
BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: ucaextendedvalidationroot
SHA1: A3:A1:B0:6F:24:61:23:4A:E3:36:A5:C2:37:FC:A6:FF:DD:F0:D7:3A
SHA256:
D4:3A:F9:B3:54:73:75:5C:96:84:FC:06:D7:D8:CB:70:EE:5C:28:E7:73:FB:29:4E:B4:1E:E7:17:22:92:4D:2
Alias name: ucaglobalg2root
SHA1: 28:F9:78:16:19:7A:FF:18:25:18:AA:44:FE:C1:A0:CE:5C:B6:4C:8A
SHA256:
9B:EA:11:C9:76:FE:01:47:64:C1:BE:56:A6:F9:14:B5:A5:60:31:7A:BD:99:88:39:33:82:E5:16:1A:A0:49:3
Alias name: usertrustecc
SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
SHA256:
4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustecccertificationauthority
SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
SHA256:
4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustrsa
SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
SHA256:
E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: usertrustrsacertificationauthority
SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
SHA256:
E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: utndatacorpsgcca
SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
SHA256:
85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: utnuserfirstclientauthemailca
```

```
SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
SHA256:
43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A
Alias name: utnuserfirsthardwareca
SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
SHA256:
6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: utnuserfirstobjectca
SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
SHA256:
6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8
Alias name: valicertclass2ca
SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
SHA256:
58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: verisignclg1.pem
SHA1: 90:AE:A2:69:85:FF:14:80:4C:43:49:52:EC:E9:60:84:77:AF:55:6F
SHA256:
D1:7C:D8:EC:D5:86:B7:12:23:8A:48:2C:E4:6F:A5:29:39:70:74:2F:27:6D:8A:B6:A9:E4:6E:E0:28:8F:33:5
Alias name: verisignclg2.pem
SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
SHA256:
34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignclg3.pem
SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
SHA256:
CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignclg6.pem
SHA1: 51:7F:61:1E:29:91:6B:53:82:FB:72:E7:44:D9:8D:C3:CC:53:6D:64
SHA256:
9D:19:0B:2E:31:45:66:68:5B:E8:A8:89:E2:7A:A8:C7:D7:AE:1D:8A:AD:DB:A3:C1:EC:F9:D2:48:63:CD:34:B
Alias name: verisignc2g1.pem
SHA1: 67:82:AA:E0:ED:EE:E2:1A:58:39:D3:C0:CD:14:68:0A:4F:60:14:2A
SHA256:
BD:46:9F:F4:5F:AA:E7:C5:4C:CB:D6:9D:3F:3B:00:22:55:D9:B0:6B:10:B1:D0:FA:38:8B:F9:6B:91:8B:2C:E
Alias name: verisignc2g2.pem
SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
SHA256:
3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignc2g3.pem
SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
SHA256:
92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignc2g6.pem
```

```
SHA1: 40:B3:31:A0:E9:BF:E8:55:BC:39:93:CA:70:4F:4E:C2:51:D4:1D:8F
SHA256:
CB:62:7D:18:B5:8A:D5:6D:DE:33:1A:30:45:6B:C6:5C:60:1A:4E:9B:18:DE:DC:EA:08:E7:DA:AA:07:81:5F:F
Alias name: verisignc3g1.pem
SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
SHA256:
A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignc3g2.pem
SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
SHA256:
83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignc3g3.pem
SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
SHA256:
EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignc3g4.pem
SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
SHA256:
69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignc3g5.pem
SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
SHA256:
9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignc4g2.pem
SHA1: 0B:77:BE:BB:CB:7A:A2:47:05:DE:CC:0F:BD:6A:02:FC:7A:BD:9B:52
SHA256:
44:64:0A:0A:0E:4D:00:0F:BD:57:4D:2B:8A:07:BD:B4:D1:DF:ED:3B:45:BA:AB:A7:6F:78:57:78:C7:01:19:6
Alias name: verisignc4g3.pem
SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
SHA256:
E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: verisignclass1ca
SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
SHA256:
51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2
Alias name: verisignclass1g2ca
SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
SHA256:
34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignclass1g3ca
SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
SHA256:
CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignclass2g2ca
```

```
SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
SHA256:
3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignclass2g3ca
SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
SHA256:
92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignclass3ca
SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
SHA256:
A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignclass3g2ca
SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
SHA256:
83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignclass3g3ca
SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
SHA256:
EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignclass3g4ca
SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
SHA256:
69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3g5ca
SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
SHA256:
9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignclass3publicprimarycertificationauthorityg4
SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
SHA256:
69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3publicprimarycertificationauthorityg5
SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
SHA256:
9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignroot.pem
SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
SHA256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisigntsaca
SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
SHA256:
CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9
Alias name: verisignuniversalrootca
```

```

SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
SHA256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisignuniversalrootcertificationauthority
SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
SHA256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: xrampglobalca
SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
SHA256:
CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: xrampglobalcaroot
SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
SHA256:
CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A

```

Verwenden Sie AWS WAF , um Ihren REST APIs in API Gateway zu schützen

AWS WAF ist eine Firewall für Webanwendungen, die zum Schutz von Webanwendungen und APIs vor Angriffen beiträgt. Sie ermöglicht es Ihnen, eine Gruppe von Regeln (eine sogenannte Web-Zugriffskontrollliste oder Web-ACL) zum Zulassen, Blockieren oder Zählen von Webanforderungen basierend auf von Ihnen definierten anpassbaren Web-Sicherheitsregeln und Bedingungen zu konfigurieren. Weitere Informationen finden Sie unter [So AWS WAF funktioniert](#) es.

Sie können AWS WAF damit Ihre API Gateway REST API vor gängigen Web-Exploits wie SQL-Injection und Cross-Site Scripting (XSS) -Angriffen schützen. Diese können sich auf die Verfügbarkeit und Leistung der API auswirken, die Sicherheit gefährden oder übermäßige Ressourcen verbrauchen. Sie können beispielsweise Regeln zum Zulassen oder Blockieren von Anforderungen von angegebenen IP-Adressbereichen, Anforderungen von CIDR-Blocks, Anforderungen aus einem bestimmten Land oder einer bestimmten Region, Anforderungen mit bösartigem SQL-Code oder Anforderungen mit bösartigem Skript erstellen.

Sie können auch Regeln erstellen, die mit einer bestimmten Zeichenfolge oder einem regulären Ausdrucksmuster in HTTP-Headern, einer Methode, einer Abfragezeichenfolge, einer URI und dem Anforderungstext (nur für die ersten 8 KB) übereinstimmen. Außerdem können Sie Regeln zum Blockieren von Angriffen von bestimmten Benutzeragenten, bösartigen Bots und Content Scrapers erstellen. Beispielsweise können Sie mit durchsatzbasierten Regeln die Anzahl der zulässigen Webanforderungen angeben, die von jeder Client-IP in einem sich anschließenden, fortlaufend aktualisierten 5-Minuten-Zeitraum zugelassen werden.

 Important

AWS WAF ist Ihre erste Verteidigungslinie gegen Web-Exploits. Wenn auf einer API aktiviert AWS WAF ist, werden AWS WAF Regeln vor anderen Zugriffskontrollfunktionen wie Ressourcenrichtlinien, IAM-Richtlinien, Lambda-Autorisierern und Amazon CognitoCognito-Autorisierern ausgewertet. Wenn beispielsweise der Zugriff über einen CIDR-Block AWS WAF blockiert wird, den eine Ressourcenrichtlinie zulässt, hat dies AWS WAF Vorrang und die Ressourcenrichtlinie wird nicht bewertet.

Um die API AWS WAF zu aktivieren, müssen Sie wie folgt vorgehen:

1. Verwenden Sie die AWS WAF Konsole, das AWS SDK oder die CLI, um eine Web-ACL zu erstellen, die die gewünschte Kombination aus AWS WAF verwalteten Regeln und Ihren eigenen benutzerdefinierten Regeln enthält. Weitere Informationen finden Sie unter [Erste Schritte mit AWS WAF](#) und [Web Access Control Lists \(Web ACLs\)](#).

 Important

API Gateway benötigt eine AWS WAFV2 Web-ACL für eine regionale Anwendung oder eine AWS WAF Classic Regional Web-ACL.

2. Ordnen Sie die AWS WAF Web-ACL einer API-Stufe zu. Sie können dies mithilfe der AWS WAF Konsole, des AWS SDK, der CLI oder der API Gateway Gateway-Konsole tun.

So verknüpfen Sie mithilfe der API Gateway Gateway-Konsole eine AWS WAF Web-ACL mit einer API-Gateway-API-Stufe

Gehen Sie wie folgt vor, um mit der API Gateway Gateway-Konsole eine AWS WAF Web-ACL einer vorhandenen API-Gateway-API-Stufe zuzuordnen:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.
3. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine Stufe aus.
4. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
5. Wählen Sie Ihre Web-ACL unter Web Application Firewall (AWS WAF) aus.

Wenn Sie verwenden AWS WAFV2, wählen Sie eine AWS WAFV2 Web-ACL für eine regionale Anwendung aus. Die Web-ACL und alle anderen AWS WAFV2 Ressourcen, die sie verwendet, müssen sich in derselben Region wie Ihre API befinden.

Wenn Sie eine regionale Web-ACL verwenden AWS WAF Classic Regional, wählen Sie eine regionale Web-ACL aus.

6. Wählen Sie **Änderungen speichern** aus.

Ordnen Sie eine AWS WAF Web-ACL einer API-Gateway-API-Stufe zu, indem Sie den AWS CLI

Der folgende [associate-web-acl](#) Befehl verknüpft eine AWS WAFV2 Web-ACL für eine regionale Anwendung mit einer vorhandenen API-Gateway-API-Stufe:

```
aws wafv2 associate-web-acl \
--web-acl-arn arn:aws:wafv2:{region}:111122223333:regional/webacl/test-cli/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
--resource-arn arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod
```

Der folgende [associate-web-acl](#) Befehl verknüpft eine AWS WAF Classic Regional Web-ACL mit einer vorhandenen API-Gateway-API-Stufe:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Ordnen Sie mithilfe der REST-API eine AWS WAF Web-ACL einer API-Stufe zu AWS WAF

Um mithilfe der AWS WAFV2 REST-API eine AWS WAFV2 Web-ACL für eine regionale Anwendung einer vorhandenen API-Gateway-API-Stufe zuzuordnen, verwenden Sie den [AssociateWebACL-Befehl](#) wie im folgenden Beispiel:

```
import boto3

wafv2 = boto3.client('wafv2')

wafv2.associate_web_acl(
```

```
WebACLArn='arn:aws:wafv2:{region}:111122223333:regional/webacl/test/abc6aa3b-
fc33-4841-b3db-0ef3d3825b25',
ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Um mithilfe der AWS WAF REST-API eine AWS WAF Classic Regional Web-ACL einer vorhandenen API-Gateway-API-Stufe zuzuordnen, verwenden Sie den [AssociateWebACL-Befehl](#) wie im folgenden Beispiel:

```
import boto3

waf = boto3.client('waf-regional')

waf.associate_web_acl(
    WebACLId='aabc123a-fb4f-4fc6-becb-2b00831cadcf',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Drosseln Sie Anfragen an Ihren REST APIs für einen besseren Durchsatz in API Gateway

Sie können Drosselung und Kontingente für Sie konfigurieren APIs, um zu verhindern, dass sie von zu vielen Anfragen überfordert werden. Sowohl Drosselungen als auch Quoten werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere untersucht API Gateway die Rate und die Anzahl der eingereichten Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise 429 Too Many Requests Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Zielgrenzwerte für einzelne API-Phasen oder Methoden festlegen, um die Gesamtleistung APIs in Ihrem Konto zu verbessern. Alternativ können Sie Nutzungspläne

aktivieren, um Drosselungen für Client-Anforderungseinreichungen basierend auf angegebenen Anforderungsraten und -kontingente einzurichten.

Themen

- [So werden Einstellungen für die Drosselungsgrenzen in API Gateway angewendet:](#)
- [Drosselung auf Kontoebene pro Region](#)
- [Konfiguration von Drosselungs-Targets auf API-Ebene und Stufenebene in einem Nutzungsplan](#)
- [Konfigurieren von Drosselungs-Targets auf Stufenebene](#)
- [Konfiguration von Drosselungs-Targets auf Methodenebene in einem Nutzungsplan](#)

So werden Einstellungen für die Drosselungsgrenzen in API Gateway angewendet:

Bevor Sie die Einstellungen für Drosselung und Kontingente für Ihre API konfigurieren, sollten Sie die Drosselungs- und Kontingenteinstellungen für Ihre API verstehen und wissen, wie API Gateway sie anwendet.

Amazon API Gateway bietet vier grundlegende Arten von Drosselungseinstellungen:

- AWS Drosselungslimits gelten für alle Konten und Kunden in einer Region. Diese Limit-Einstellungen verhindern, dass Ihre API – und Ihr Konto – von zu vielen Anfragen überfordert wird. Diese Limits werden von einem Kunden festgelegt AWS und können von diesem nicht geändert werden.
- Limits pro Konto gelten für alle APIs Konten in einer bestimmten Region. Das Ratenlimit auf Kontoebene kann auf Anfrage erhöht werden. Höhere Limits sind möglich, was kürzere Timeouts und geringere Payloads zur Folge hat. APIs Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das [AWS Supportcenter](#). Weitere Informationen finden Sie unter [Kontingente und wichtige Hinweise](#). Beachten Sie, dass diese Limits nicht höher als die Drosselungsgrenzen sein können. AWS
- Die Drosselungsgrenzen pro API pro Stufe werden auf API-Methodenebene für eine Stufe angewendet. Sie können für alle Methoden dieselben Einstellungen konfigurieren oder für jede Methode unterschiedliche Drosseleinstellungen konfigurieren. Beachten Sie, dass diese Grenzwerte nicht höher als die AWS Drosselungsgrenzen sein dürfen.
- Client-basierte Drossel-Limits werden auf Clients angewendet, die mit Ihrem Nutzungsplan verknüpfte API-Schlüssel als Client-ID verwenden. Beachten Sie, dass diese Limits nicht höher sein dürfen als die Grenzwerte pro Konto.

API Gateway wendet Ihre Drosselungseinstellungen in der folgenden Reihenfolge an:

1. [Client-basierte Drossel-Limits oder Drossel-Limits pro Methode](#), die Sie im [Nutzungsplan](#) für eine API-Stufe festlegen
2. [Drossel-Limits pro Methode, die Sie für eine API-Stufe festlegen](#)
3. [Drosselung auf Kontoebene pro Region](#)
4. AWS Regionale Drosselung

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) APIs innerhalb eines AWS Kontos pro Region. Außerdem wird der Burst (d. h. die maximale Bucket-Größe) für alle APIs innerhalb eines AWS Kontos pro Region begrenzt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von 429 Too Many Requests-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter [Kontingente und wichtige Hinweise](#).

Konfiguration von Drosselungs-Targets auf API-Ebene und Stufenebene in einem Nutzungsplan

In einem [Nutzungsplan](#) können Sie ein Drosselungs-Target pro Methode für alle Methoden auf API- oder Stufenebene festlegen. Sie können eine Drosselungsrate angeben, d. h. die Rate in Anfragen pro Sekunde, mit der Token zum Token-Bucket hinzugefügt werden. Sie können auch einen Drosselungs-Burst angeben, der der Kapazität des Token-Buckets entspricht.

Sie können die AWS CLI SDKs, und die verwenden, AWS-Managementkonsole um einen Nutzungsplan zu erstellen. Informationen zum Erstellen eines Nutzungsplans finden Sie unter [???](#).

Konfigurieren von Drosselungs-Targets auf Stufenebene

Sie können die AWS CLI, und die verwenden SDKs, AWS-Managementkonsole um Drosselungsziele auf Stufenebene zu erstellen.

Weitere Informationen zur Verwendung von zum Erstellen von Drosselungszielen AWS-Managementkonsole auf Stufenebene finden Sie unter. [??? Weitere Informationen zur Verwendung der AWS CLI zum Erstellen von Drosselungszielen auf Stufenebene finden Sie unter create-stage.](#)

Konfiguration von Drosselungs-Targets auf Methodenebene in einem Nutzungsplan

Sie können weitere Drosselungs-Targets auf Methodenebene in Usage Plans (Nutzungsplänen) festlegen, wie unter [Erstellen eines Nutzungsplans](#) dargestellt. In der API Gateway-Konsole werden diese durch die Angabe von Resource=<resource>, Method=<method> in der Einstellung Configure Method Throttling (Methodendrosselung konfigurieren) festgelegt. [Für dieses Beispiel könnten Sie beispielsweise, angebenPetStore.](#) Resource=/pets Method=GET

Privates REST APIs im API Gateway

Eine private API ist eine REST-API, die nur aus einer Amazon-VPC heraus aufgerufen werden kann. Sie können einen [Schnittstellen-VPC-Endpunkt](#) zum Aufruf Ihrer API verwenden, bei dem es sich um eine in Ihrem VPC erstellte Netzwerkschnittstelle für den Endpunkt handelt. Schnittstellenendpunkte werden von einer Technologie unterstützt AWS PrivateLink, die es Ihnen ermöglicht, privat auf AWS Dienste zuzugreifen, indem Sie private IP-Adressen verwenden.

Sie können es auch verwenden Direct Connect , um eine Verbindung von einem lokalen Netzwerk zu Amazon VPC herzustellen und dann über diese Verbindung auf Ihre private API zuzugreifen. In allen Fällen verwendet der Datenverkehr zu Ihrer privaten API eine sichere Verbindung, vollkommen isoliert vom öffentlichen Internet. Der Datenverkehr verlässt das Amazon-Netzwerk nicht.

Bewährte Methoden für Privatanwender APIs

Wir empfehlen die Verwendung der folgenden bewährten Methoden zur Erstellung Ihrer privaten API:

- Verwenden Sie einen einzigen VPC-Endpunkt, um auf mehrere private APIs Endpunkte zuzugreifen. Dadurch verringert sich die Anzahl der eventuell erforderlichen VPC-Endpunkte.
- Ordnen Sie Ihren VPC-Endpunkt Ihrer API zu. Dadurch wird ein Route-53-Alias-DNS-Datensatz erstellt, der das Aufrufen Ihrer privaten API vereinfacht.
- Aktivieren Sie ein privates DNS für Ihre VPC. Wenn Sie ein privates DNS für Ihre VPC aktivieren, können Sie Ihre API innerhalb der VPC aufrufen, ohne den Host- oder x-apigw-api-id-Header übergeben zu müssen.

Wenn Sie privates DNS aktivieren, können Sie nicht auf den Standardendpunkt für öffentlich APIs zugreifen. Um auf den Standardendpunkt für öffentlich zugreifen APIs, können Sie privates DNS deaktivieren, eine private gehostete Zone für jede private API in Ihrer VPC erstellen und dann die erforderlichen Datensätze in Route 53 bereitstellen. Auf diese Weise kann Ihre private API aufgelöst werden, während Sie weiterhin den öffentlichen Standardendpunkt von Ihrer VPC aus aufrufen können. Weitere Informationen finden Sie unter [Erstellen einer privat gehosteten Zone](#).

- Beschränken Sie den Zugriff auf Ihre private API auf bestimmte VPCs oder VPC-Endpunkte. Fügen Sie die Bedingung `aws:SourceVpc` oder `aws:SourceVpce` zu Ihrer API-Ressourcenrichtlinie hinzu, um Zugriff einzuschränken.
- Den sichersten Datenperimeter garantieren Sie mit einer VPC-Endpunktrichtlinie. Sie steuert den Zugriff auf die VPC-Endpunkte, die Ihre private API aufrufen können.

Überlegungen zu privaten APIs

Die folgenden Überlegungen könnten sich auf Ihre Nutzung von Private auswirken APIs:

- Es APIs werden nur REST unterstützt.
- Sie können keine private API in eine Edge-optimierte API konvertieren.
- Private unterstützt APIs nur TLS 1.2. Frühere TLS-Versionen werden nicht unterstützt.
- Wenn Sie eine Anfrage über das HTTP/2-Protokoll stellen, wird die Anfrage gezwungen, das HTTP/1.1-Protokoll zu verwenden.
- Sie können den IP-Adresstyp nicht APIs auf privat festlegen, sodass nur IPv4 Adressen Ihre private API aufrufen können. Nur Dualstack wird unterstützt. Weitere Informationen finden Sie unter [the section called “IP-Adresstypen für REST-APIs in API Gateway”](#).
- Um Datenverkehr über Ihre private API zu senden, können Sie alle von Amazon VPC unterstützten IP-Adresstypen verwenden. Sie können Dualstack und IPv6 Traffic senden, indem Sie die Einstellungen auf Ihrem VPC-Endpunkt konfigurieren. In API Gateway kann dies nicht verändert werden. Weitere Informationen finden [Sie unter IPv6 Unterstützung für Ihre VPC hinzufügen](#).
- VPC-Endpunkte für private Geräte APIs unterliegen denselben Einschränkungen wie andere VPC-Endpunkte mit Schnittstellen. Weitere Informationen finden Sie im AWS PrivateLink Handbuch unter [Zugreifen auf einen AWS Dienst über einen Schnittstellen-VPC-Endpunkt](#). Weitere Informationen zur Verwendung von API Gateway mit gemeinsam genutzten VPCs und geteilten Subnetzen finden Sie im Handbuch unter [Gemeinsam genutzte Subnetze](#).AWS PrivateLink

Die nächsten Schritte für private APIs

Anleitungen zum Erstellen einer privaten API und zum Zuordnen eines VPC-Endpunkts finden Sie unter [the section called “Erstellen einer privaten API”](#). Eine Anleitung, in der Sie Abhängigkeiten CloudFormation und eine private API in der erstellen AWS-Managementkonsole, finden Sie unter [the section called “Tutorial: Erstellen einer privaten REST-API”](#).

Erstellen einer privaten API

Bevor Sie eine private API erstellen, brauchen Sie zunächst einen VPC-Endpunkt für API Gateway. Als Nächstes erstellen Sie Ihre private API und fügen ihr eine Ressourcenrichtlinie hinzu. Optional können Sie Ihren VPC-Endpunkt mit Ihrer privaten API verknüpfen, um den Aufruf Ihrer API zu vereinfachen. Im letzten Schritt stellen Sie Ihre API bereit.

Die entsprechende Vorgehensweise wird nachfolgend beschrieben. Sie können eine private REST-API mit der AWS-Managementkonsole, AWS CLI oder einem AWS SDK erstellen.

Voraussetzungen

Sie brauchen eine vollständig konfigurierte VPC, um diese Schritte ausführen zu können. Informationen zum Erstellen einer VPC finden Sie unter [Erstellen einer VPC](#) im Amazon-VPC-Benutzerhandbuch. Aktivieren Sie Ihr privates DNS, um allen empfohlenen Schritten für die Erstellung Ihrer VPC folgen zu können. Auf diese Weise können Sie Ihre API innerhalb einer VPC aufrufen, ohne den Host oder den `x-api-gw-api-id`-Header übergeben zu müssen.

Die Attribute `enableDnsSupport` und `enableDnsHostnames` Ihrer VPC müssen auf `true` gesetzt sein, um das private DNS aktivieren zu können. Weitere Informationen finden Sie unter [DNS-Support für Ihre VPC](#) und [Aktualisieren des DNS-Supports für Ihre VPC](#).

Schritt 1: Erstellen eines VPC-Endpunkts für API Gateway in Ihrer VPC

Im Folgenden sehen Sie, wie Sie einen VPC-Endpunkt für API Gateway erstellen. Um einen VPC-Endpunkt für API Gateway zu erstellen, geben Sie die `execute-api` Domain für die Domain an, AWS-Region in der Sie Ihre private API erstellen. Die `execute-api`-Domain ist der API-Gateway-Komponenten-Service für die API-Ausführung.

Beim Erstellen des VPC-Endpunkts für API Gateway legen Sie auch die DNS-Einstellungen fest. Wenn Sie das private DNS deaktivieren, können Sie nur über ein öffentliches DNS auf Ihre API zugreifen. Weitere Informationen finden Sie unter [the section called “Problem: Ich kann keine Verbindung zwischen einem API-Gateway-VPC-Endpunkt und meiner öffentlichen API herstellen”](#).

AWS-Managementkonsole

So erstellen Sie einen Schnittstellen-VPC-Endpunkt für API Gateway

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon VPC-Konsole unter <https://console.aws.amazon.com/vpc/>.

2. Wählen Sie im Navigationsmenü unter Virtual Private Cloud die Option Endpunkte.
3. Wählen Sie Endpunkt erstellen aus.
4. (Optional) Geben Sie für Name-Tag einen Namen ein, um Ihren VPC-Endpunkt zu identifizieren.
5. Wählen Sie bei Service category (Servicekategorie) die Option AWS services (-Services) aus.
6. Geben Sie im Suchfeld Services **execute-api** ein. Wählen Sie dann den API Gateway Gateway-Dienstendpunkt aus, auf AWS-Region dem Sie Ihre API erstellen möchten. Der Dienstname sollte wie folgt aussehen: `com.amazonaws.us-east-1.execute-api`, und der Typ sollte Interface sein.
7. Wählen Sie für VPC die VPC aus, in der Sie den Endpunkt erstellen möchten.
8. (Optional) Wenn Sie Privaten DNS-Namen aktivieren deaktivieren wollen, wählen Sie Zusätzliche Einstellungen aus und löschen Sie den Feldinhalt für Privaten DNS-Namen aktivieren.
9. Wählen Sie für Subnetze die Availability Zones aus, in denen Sie die Netzwerkschnittstellen für die Endpunkte erstellt haben. Zur Verbesserung der Verfügbarkeit Ihrer API wählen Sie mehrere Subnetze aus.
10. Wählen Sie für Security group (Sicherheitsgruppe) die Sicherheitsgruppe aus, die den VPC-Endpunktnetzwerkschnittstellen zugeordnet werden soll.

Die von Ihnen ausgewählte Sicherheitsgruppe muss so konfiguriert sein, dass eingehender HTTPS-Datenverkehr über TCP-Port 443 von einem IP-Adressbereich in Ihrer VPC oder einer anderen Sicherheitsgruppe in Ihrer VPC zugelassen wird.

11. Führen Sie für Richtlinie einen der folgenden Schritte aus:
 - Wenn Sie Ihre private API nicht erstellt haben oder keine benutzerdefinierte VPC-Endpunktrichtlinie konfigurieren möchten, wählen Sie Vollzugriff aus.
 - Wenn Sie bereits eine private API erstellt haben und eine benutzerdefinierte VPC-Endpunktrichtlinie konfigurieren möchten, geben Sie eine benutzerdefinierte VPC-Endpunktrichtlinie ein. Weitere Informationen finden Sie unter [the section called “VPC-Endpunktrichtlinien für private Zwecke verwenden APIs”](#).

Sie können die VPC-Endpunktrichtlinie nach dem Erstellen des VPC-Endpunkts aktualisieren. Weitere Informationen finden Sie unter [Aktualisieren einer VPC-Endpunktrichtlinie](#).

12. Wählen Sie Endpunkt erstellen aus.
13. Kopieren Sie die resultierende VPC-Endpunkt-ID, da Sie sie u. U. bei späteren Schritten noch brauchen werden.

AWS CLI

Der folgende [create-vpc-endpoint](#) Befehl erstellt einen VPC-Endpoint:

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --vpc-endpoint-type Interface \  
  --service-name com.amazonaws.us-east-1.execute-api \  
  --subnet-ids subnet-7b16de0c \  
  --security-group-id sg-1a2b3c4d
```

Kopieren Sie die resultierende VPC-Endpoint-ID, da Sie sie u. U. bei späteren Schritten noch brauchen werden.

Schritt 2: Erstellen einer privaten API

Nachdem Sie Ihren VPC-Endpoint erstellt haben, erstellen Sie nun eine private REST-API. Im folgenden Verfahren sehen Sie, wie Sie eine private API erstellen.

AWS-Managementkonsole

Erstellen einer privaten API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Create API (API erstellen) aus.
3. Wählen Sie unter REST-API die Option Build (Erstellen) aus.
4. Geben Sie unter Name einen Namen ein.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
6. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
7. (Optional) Geben Sie für VPC-Endpoint IDs eine VPC-Endpoint-ID ein.

Wenn Sie Ihrer privaten API eine VPC-Endpoint-ID zuordnen, können Sie Ihre API von Ihrer VPC aus aufrufen, ohne einen Host-Header zu überschreiben oder einen `x-apigw-api-id` header zu übergeben. Weitere Informationen finden Sie unter [the section called “\(Optional\) Zuordnen oder Aufheben der Zuweisung eines VPC-Endpunkts zu einer privaten API”](#).

8. Wählen Sie bei IP address type (IP-Adresstyp) Dualstack.
9. Wählen Sie Create API (API erstellen) aus.

Nachdem Sie die Schritte oben abgeschlossen haben, können Sie der Anleitung unter [the section called “Erste Schritte mit der REST-API-Konsole”](#) folgen, um Methoden und Integrationen für diese API einzurichten - Sie können die API jedoch noch nicht bereitstellen. Zur Bereitstellung Ihrer API folgen Sie Schritt 3 und fügen Ihrer API eine Ressourcenrichtlinie hinzu.

AWS CLI

Der folgende [create-rest-api](#) Befehl erstellt eine private API:

```
aws apigateway create-rest-api \  
    --name 'Simple PetStore (AWS CLI, Private)' \  
    --description 'Simple private PetStore API' \  
    --region us-west-2 \  
    --endpoint-configuration '{ "types": ["PRIVATE"], "ipAddressType":  
    "dualstack" }'
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
{  
  "createdDate": "2017-10-13T18:41:39Z",  
  "description": "Simple private PetStore API",  
  "endpointConfiguration": {  
    "types": [  
      "PRIVATE"  
    ],  
    "ipAddressType": "dualstack"  
  },  
  "id": "0qzs2sy7bh",  
  "name": "Simple PetStore (AWS CLI, Private)"  
}
```

Nachdem Sie die Schritte oben abgeschlossen haben, können Sie der Anleitung unter [the section called “Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI”](#) folgen, um Methoden und Integrationen für diese API einzurichten - Sie können die API jedoch noch nicht bereitstellen. Zur Bereitstellung Ihrer API folgen Sie Schritt 3 und fügen Ihrer API eine Ressourcenrichtlinie hinzu.

SDK JavaScript v3

Das folgende Beispiel zeigt, wie eine private API mithilfe des AWS SDK für JavaScript Version 3 erstellt wird:

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
const apig = new APIGatewayClient({region:"us-east-1"});

const input = { // CreateRestApiRequest
  name: "Simple PetStore (JavaScript v3 SDK, private)", // required
  description: "Demo private API created using the AWS SDK for JavaScript v3",
  version: "0.00.001",
  endpointConfiguration: { // EndpointConfiguration
    types: [ "PRIVATE"],
  },
};

export const handler = async (event) => {
  const command = new CreateRestApiCommand(input);
  try {
    const result = await apig.send(command);
    console.log(result);
  } catch (err){
    console.error(err)
  }
};
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
{
  apiKeySource: 'HEADER',
  createdAt: 2024-04-03T17:56:36.000Z,
  description: 'Demo private API created using the AWS SDK for JavaScript v3',
  disableExecuteApiEndpoint: false,
  endpointConfiguration: { types: [ 'PRIVATE' ] },
  id: 'abcd1234',
  name: 'Simple PetStore (JavaScript v3 SDK, private)',
  rootResourceId: 'efg567',
  version: '0.00.001'
}
```

Nachdem Sie die Schritte oben abgeschlossen haben, können Sie der Anleitung unter [the section called “Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI”](#) folgen, um Methoden und Integrationen für diese API einzurichten - Sie können die API jedoch noch nicht bereitstellen. Zur Bereitstellung Ihrer API folgen Sie Schritt 3 und fügen Ihrer API eine Ressourcenrichtlinie hinzu.

Python SDK

Das folgende Beispiel zeigt, wie Sie mithilfe des AWS SDK für Python eine private API erstellen:

```
import json
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

def lambda_handler(event, context):
    try:
        result = apig.create_rest_api(
            name='Simple PetStore (Python SDK, private)',
            description='Demo private API created using the AWS SDK for Python',
            version='0.00.001',
            endpointConfiguration={
                'types': [
                    'PRIVATE',
                ],
            },
        )
    except botocore.exceptions.ClientError as error:
        logger.exception("Couldn't create private API %s.", error)
        raise
    attribute=["id", "name", "description", "createdDate", "version",
"apiKeySource", "endpointConfiguration"]
    filtered_data={key:result[key] for key in attribute}
    result = json.dumps(filtered_data, default=str, sort_keys='true')
    return result
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
{"apiKeySource\": \"HEADER\", \"createdDate\": \"2024-04-03 17:27:05+00:00\",
\"description\": \"Demo private API created using the AWS SDK for \",
\"endpointConfiguration\": {\"types\": [\"PRIVATE\"]}, \"id\": \"abcd1234\", \"name
\": \"Simple PetStore (Python SDK, private)\", \"version\": \"0.00.001\"}"
```

Nachdem Sie die Schritte oben abgeschlossen haben, können Sie der Anleitung unter [the section called “Tutorial: Erstellen Sie eine REST-API mit AWS SDKs oder AWS CLI”](#) folgen, um Methoden und Integrationen für diese API einzurichten - Sie können die API jedoch noch nicht bereitstellen.

Zur Bereitstellung Ihrer API folgen Sie Schritt 3 und fügen Ihrer API eine Ressourcenrichtlinie hinzu.

Schritt 3: Einrichten einer Ressourcenrichtlinie für eine private API

Ihre aktuelle private API ist für alle VPCs nicht zugänglich. Verwenden Sie eine Ressourcenrichtlinie, um Ihren VPCs und VPC-Endpunkten Zugriff auf Ihre privaten Daten zu gewähren. APIs Sie können in jedem AWS Konto Zugriff auf einen VPC-Endpunkt gewähren.

Ihre Ressourcenrichtlinie sollte `aws:SourceVpc-` oder `aws:SourceVpce-` Bedingungen für Zugriffsbeschränkungen enthalten. Wir empfehlen, dass Sie spezifische VPCs und VPC-Endpunkte identifizieren und keine Ressourcenrichtlinie erstellen, die den Zugriff für alle VPCs VPC-Endpunkte ermöglicht.

Im folgenden Verfahren sehen Sie, wie Sie eine Ressourcenrichtlinie an Ihre API anfügen.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Wählen Sie Richtlinie erstellen aus.
5. Wählen Sie Vorlage auswählen und dann Quell-VPC aus.
6. Ersetzen Sie `{{vpcID}}` (einschließlich der geschweiften Klammern) durch Ihre VPC-ID.
7. Wählen Sie Änderungen speichern aus.

AWS CLI

Mit dem folgenden [update-rest-api](#) Befehl wird eine Ressourcenrichtlinie an eine bestehende API angehängt:

```
aws apigateway update-rest-api \  
  --rest-api-id a1b2c3 \  
  --patch-operations op=replace,path=/  
policy,value='{"jsonEscapedPolicyDocument"}'
```

Sie sollten u. U. auch vorgeben, welche Ressourcen Zugriff auf Ihren VPC-Endpunkt haben sollen. Verbinden Sie eine VPC-Endpunkttrichtlinie mit Ihrem VPC-Endpunkt, um festzulegen, welche Ressourcen Zugriff auf Ihren VPC-Endpunkt haben sollen. Weitere Informationen finden Sie unter [the section called “VPC-Endpunkttrichtlinien für private Zwecke verwenden APIs”](#).

(Optional) Zuordnen oder Aufheben der Zuweisung eines VPC-Endpunkts zu einer privaten API

Wenn Sie einen VPC-Endpunkt mit Ihrer privaten API verknüpfen, generiert API Gateway einen neuen Route-53-ALIAS-DNS-Eintrag. Sie können diesen Datensatz verwenden, um Ihren privaten Datensatz APIs genauso aufzurufen, wie Sie Ihren öffentlichen Eintrag aufrufen, APIs ohne einen Header zu überschreiben oder einen Host Header zu übergeben. `x-apigw-api-id`

Die generierte Basis-URL liegt im folgenden Format vor:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (AWS-Managementkonsole)

Sie können Ihrer privaten API bei oder nach der Erstellung einen VPC-Endpunkt zuordnen. Im folgenden Verfahren sehen Sie, wie Sie einen VPC-Endpunkt mit einer zuvor erstellten API verknüpfen.

Zuordnen eines VPC-Endpunkts zu einer privaten API

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre private API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Bearbeiten Sie Ihre Ressourcenrichtlinie, um Anrufe von Ihrem zusätzlichen VPC-Endpunkt zuzulassen.
5. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
6. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
7. Wählen Sie für VPC-Endpunkt IDs einen zusätzlichen VPC-Endpunkt aus. IDs
8. Wählen Sie Speichern.
9. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Dissociate a VPC endpoint (AWS-Managementkonsole)

Die Zuordnung eines VPC-Endpunkts zu einer privaten REST-API entfernen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre private API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Bearbeiten Sie Ihre Ressourcenrichtlinie, um Erwähnungen des VPC-Endpunkts zu entfernen, den Sie von Ihrer privaten API trennen möchten.
5. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
6. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
7. Wählen Sie für VPC-Endpunkt das X aus IDs, um den VPC-Endpunkt zu trennen.
8. Wählen Sie Speichern.
9. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Associate a VPC endpoint (AWS CLI)

Der folgende [create-rest-api](#)Befehl ordnet VPC-Endpunkte zum Zeitpunkt der API-Erstellung zu:

```
aws apigateway create-rest-api \  
  --name Petstore \  
  --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :  
  ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \  
  --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "apiKeySource": "HEADER",  
  "endpointConfiguration": {  
    "types": [  
      "PRIVATE"  
    ],  
    "vpcEndpointIds": [  
      "vpce-0212a4ababd5b8c3e",  
      "vpce-0393a628149c867ee"  
    ]  
  }  
}
```

```
{,
  "id": "u67n3ov968",
  "createdDate": 1565718256,
  "name": "Petstore"
}
```

Der folgende [update-rest-api](#) Befehl ordnet VPC-Endpoints einer API zu, die Sie bereits erstellt haben:

```
aws apigateway update-rest-api \
  --rest-api-id u67n3ov968 \
  --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
  --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{
  "name": "Petstore",
  "apiKeySource": "1565718256",
  "tags": {},
  "createdDate": 1565718256,
  "endpointConfiguration": {
    "vpcEndpointIds": [
      "vpce-0212a4ababd5b8c3e",
      "vpce-0393a628149c867ee",
      "vpce-01d622316a7df47f9"
    ],
    "types": [
      "PRIVATE"
    ]
  },
  "id": "u67n3ov968"
}
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Disassociate a VPC endpoint (AWS CLI)

Der folgende [update-rest-api](#) Befehl trennt einen VPC-Endpunkt von einer privaten API:

```
aws apigateway update-rest-api \
```



```
--rest-api-id u67n3ov968 \  
--patch-operations "op='remove',path='/endpointConfiguration/  
vpcEndpointIds',value='vpce-0393a628149c867ee'" \  
--region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "name": "Petstore",  
  "apiKeySource": "1565718256",  
  "tags": {},  
  "createdDate": 1565718256,  
  "endpointConfiguration": {  
    "vpcEndpointIds": [  
      "vpce-0212a4ababd5b8c3e",  
      "vpce-01d622316a7df47f9"  
    ],  
    "types": [  
      "PRIVATE"  
    ]  
  },  
  "id": "u67n3ov968"  
}
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Schritt 4: Bereitstellen einer privaten API

Erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Stufe. Im folgenden Verfahren sehen Sie, wie Sie Ihre private API bereitstellen.

AWS-Managementkonsole

Bereitstellen einer privaten API

1. Wählen Sie Ihre API aus.
2. Klicken Sie auf Deploy API.
3. Wählen Sie für Stufe die Option Neue Stufe aus.
4. Geben Sie unter Stage name (Stufenname) einen Namen für die Stufe ein.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.

6. Wählen Sie Bereitstellen.

AWS CLI

Der folgende [create-deployment](#)-Befehl stellt eine private API bereit:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \  
  --stage-name test \  
  --stage-description 'Private API test stage' \  
  --description 'First deployment'
```

Fehlerbehebung für Ihre private API

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Erstellung einer privaten API auftreten können.

Problem: Ich kann keine Verbindung zwischen einem API-Gateway-VPC-Endpunkt und meiner öffentlichen API herstellen

Beim Erstellen Ihrer VPC haben Sie die Möglichkeit, die DNS-Einstellungen zu konfigurieren. Wir empfehlen, Ihr privates DNS für Ihre VPC zu aktivieren. Wenn Sie das private DNS deaktivieren, ist ein Zugriff auf Ihr API nur über das öffentliche DNS möglich.

Wenn Sie Ihr privates DNS aktivieren, können Sie von Ihrem VPC-Endpunkt aus nicht auf den Standardendpunkt einer öffentlichen API-Gateway-API zugreifen. Sie können auf eine API mit einem benutzerdefinierten Domainnamen zugreifen.

Wenn Sie einen benutzerdefinierten regionalen Domainnamen erstellen, verwenden Sie einen Aliaseintrag vom Typ A. Wenn Sie einen Edge-optimierten benutzerdefinierten Domainnamen erstellen, bestehen keine Einschränkungen für Ihren Datensatztyp. Sie können auf diese öffentlichen Bereiche zugreifen, wenn privates DNS APIs aktiviert ist. Weitere Informationen finden Sie unter [Problem: Verbindungsaufbau mit meiner öffentlichen API von einem API-Gateway-VPC-Endpunkt](#).

Problem: Meine API gibt **{"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:*****/****/****/"}** zurück

Wenn Sie in Ihrer Ressourcenrichtlinie den Principal auf einen AWS Prinzipal festlegen, wie zum Beispiel:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/developer",
          "arn:aws:iam::111122223333:role/Admin"
        ]
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*stage/GET/pets"
      ]
    }
  ]
}
```

Dann müssen Sie die AWS_IAM-Genehmigung für jede einzelne Methode in Ihrer API verwenden, andernfalls gibt Ihre API die o. g. Fehlermeldung zurück. Weitere Anweisungen zum Aktivieren der AWS_IAM-Genehmigung für eine Methode finden Sie unter [the section called “Methoden”](#).

Problem: Ich bin mir nicht sicher, ob mein VPC-Endpunkt mit meiner API verknüpft ist

Wenn Sie einen VPC-Endpunkt mit Ihrer privaten API verknüpfen oder die Zuweisung aufheben, müssen Sie Ihre API erneut bereitstellen. Der Aktualisierungsvorgang kann aufgrund der DNS-Verbreitung einige Minuten dauern. Während dieser Zeit ist Ihre API verfügbar, aber die DNS-Propagierung für das neu generierte DNS ist URLs möglicherweise noch im Gange. Wenn Ihre neuen nach einigen Minuten nicht im DNS aufgelöst URLs werden, empfehlen wir Ihnen, Ihre API erneut bereitzustellen.

Benutzerdefinierte Domainnamen für private Zwecke APIs in API Gateway

Sie können einen benutzerdefinierten Domainnamen für Ihre private Domain erstellen APIs. Mit einem privaten benutzerdefinierten Domainnamen bieten Sie API-Aufrufern eine einfachere und intuitivere URL. Mit einem privaten benutzerdefinierten Domainnamen können Sie die Komplexität reduzieren, Sicherheitsmaßnahmen während des TLS-Handshakes konfigurieren

und den Zertifikatslebenszyklus Ihres Domainnamens mithilfe von AWS Certificate Manager (ACM) kontrollieren. Weitere Informationen finden Sie unter [the section called “Den privaten Zertifikatsschlüssel für Ihren benutzerdefinierten Domainnamen sichern”](#).

Benutzerdefinierte Domainnamen für private APIs Zwecke müssen nicht für mehrere Konten eindeutig sein. Sie können `example.private.com` für Konto 111122223333 und für Konto 5555555555 erstellen, vorausgesetzt, dass Ihr ACM-Zertifikat den Domainnamen abdeckt. Verwenden Sie den privaten benutzerdefinierten Domainnamen-ARN zur Identifizierung eines privaten benutzerdefinierten Domainnamens. Diese Kennung gilt nur für private benutzerdefinierte Domainnamen.

Wenn Sie in API Gateway einen privaten benutzerdefinierten Domainnamen erstellen, sind Sie ein API-Anbieter. Sie können Ihren privaten benutzerdefinierten Domainnamen AWS-Konten mithilfe von API Gateway oder AWS Resource Access Manager (AWS RAM) anderen zur Verfügung stellen.

Wenn Sie einen privaten benutzerdefinierten Domainnamen aufrufen, sind Sie ein API-Verbraucher. Sie können einen privaten benutzerdefinierten Domainnamen von Ihrem eigenen AWS-Konto oder einem anderen verwenden AWS-Konto.

Wenn Sie einen privaten benutzerdefinierten Domainnamen verbrauchen, erstellen Sie eine Domainnamenzugriffszuweisung zwischen einem VPC-Endpunkt und einem privaten benutzerdefinierten Domainnamen. Mit einer Domainnamenzugriffszuweisung können API-Verbraucher Ihren privaten benutzerdefinierten Domainnamen aufrufen, während sie vom öffentlichen Internet isoliert sind. Weitere Informationen finden Sie unter [the section called “API-Anbieter und API-Verbraucher”](#).

Den privaten Zertifikatsschlüssel für Ihren benutzerdefinierten Domainnamen sichern

Wenn Sie mit ACM ein SSL/TLS Zertifikat anfordern, um Ihren benutzerdefinierten Domainnamen für private Zwecke zu erstellen APIs, generiert ACM ein public/private key pair. Das Schlüsselpaar wird beim Import eines Zertifikats generiert. Der öffentliche Schlüssel wird Teil des Zertifikats. Um den privaten Schlüssel sicher zu speichern, erstellt ACM mithilfe AWS KMS des so genannten KMS-Schlüssels einen weiteren Schlüssel mit dem Alias `aws/acm`. AWS KMS verwendet diesen Schlüssel, um den privaten Schlüssel Ihres Zertifikats zu verschlüsseln. Weitere Informationen finden Sie unter [Datenschutz im AWS Certificate Manager](#) des AWS Certificate Manager -Benutzerhandbuchs.

API Gateway verwendet AWS TLS Connection Manager, einen Dienst, auf den nur zugegriffen werden kann AWS-Services, um die privaten Schlüssel Ihres Zertifikats zu sichern und zu verwenden. Wenn Sie Ihr ACM-Zertifikat verwenden, um einen benutzerdefinierten API Gateway-

Domännennamen zu erstellen, ordnet API Gateway Ihr Zertifikat dem AWS TLS Connection Manager zu. Wir tun dies, indem wir einen Zuschuss für Ihren AWS KMS AWS verwalteten Schlüssel erstellen. Dieser Zuschuss ermöglicht es dem TLS Connection Manager AWS KMS, den privaten Schlüssel Ihres Zertifikats zu entschlüsseln. TLS Connection Manager verwendet das Zertifikat und den entschlüsselten privaten (Klartext) Schlüssel für den Aufbau einer sicheren Verbindung (SSL/TLS-Sitzung) mit Kunden von API-Gateway-Diensten. Wenn die Zuweisung eines Zertifikats von einem API-Gateway-Service getrennt wird, wird die Berechtigung zurückgezogen. Weitere Informationen finden Sie unter [Berechtigungen](#) im AWS Key Management Service -Entwicklerhandbuch.

Weitere Informationen finden Sie unter [the section called "Verschlüsselung im Ruhezustand"](#).

Überlegungen zu privaten benutzerdefinierten Domainnamen

Die folgenden Überlegungen können Ihre Verwendung von benutzerdefinierten privaten Domainnamen beeinflussen:

- Es dauert etwa 15 Minuten, bis API Gateway Ihren privaten benutzerdefinierten Domainnamen bereitstellt.
- Wenn Sie Ihr ACM-Zertifikat aktualisieren, benötigt API Gateway etwa 15 Minuten, um die Aktualisierung abzuschließen. Während dieser Zeit befindet sich Ihr Domainname im Status UPDATING und Sie können weiterhin darauf zugreifen.
- Sie müssen eine Domainnamenzugriffszuweisung erstellen, um einen privaten benutzerdefinierten Domainnamen aufzurufen. Nach der Erstellung der Domainnamenzugriffszuweisung dauert es etwa 15 Minuten, bis diese bereitsteht.
- Der private benutzerdefinierte Domainname ARN enthält die *account-id* und die *domain-name-id*. Wenn Sie einen Domainnamen erstellen, verwendet API Gateway das ARN-Format `arn:partition:apigateway:region::/domainnames/domain-name`. Wenn Sie auf einen privaten benutzerdefinierten Domainnamen zugreifen, verwenden Sie das ARN-Format `arn:arn:partition:apigateway:region:account-id:/domainnames/domain-name+domain-name-id`.

Möglicherweise müssen Sie Ihre IAM-Berechtigungen anpassen, um nach dem Erstellen den Zugriff auf einen privaten Domainnamen zu ermöglichen.

- Sie können gleichnamige private benutzerdefinierte Domainnamen nicht vom selben VPC-Endpunkt aus aufrufen. Wenn Sie beispielsweise `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234` und `arn:aws:apigateway:us-west-2:111122223333:/domainnames/`

`private.example.com+xyz000` aufrufen möchten, müssen die einzelnen privaten benutzerdefinierten Domainnamen jeweils einem anderen VPC-Endpunkt zugeordnet werden.

- Wildcard-Zertifikate werden unterstützt, z. B. ein Zertifikat für `*.private.example.com`.
- Unterstützung für benutzerdefinierte Wildcard-Domainnamen werden nicht unterstützt.
- Es werden ausschließlich RSA-Zertifikate mit einer Schlüssellänge von 2048 Bit und ECDSA-Zertifikate mit einer Schlüssellänge von 256 Bit und 384 Bit unterstützt.
- Sie können den IP-Adresstyp nicht APIs auf privat festlegen, sodass nur IPv4 Adressen Ihre private API aufrufen können. Nur Dualstack wird unterstützt. Weitere Informationen finden Sie unter [the section called “IP-Adresstypen für REST-APIs in API Gateway”](#).
- Um Datenverkehr über Ihre private API zu senden, können Sie alle von Amazon VPC unterstützten IP-Adresstypen verwenden. Sie können Dualstack und IPv6 Traffic senden, indem Sie die Einstellungen auf Ihrem VPC-Endpunkt konfigurieren. In API Gateway kann dies nicht verändert werden. Weitere Informationen finden [Sie unter IPv6 Unterstützung für Ihre VPC hinzufügen](#).
- Mehrstufige Basispfadzuweisungen, wie etwa die Zuweisung Ihrer privaten API an `/developers/feature`, werden nicht unterstützt. Sie können jedoch eine Routing-Regel verwenden, um eine mehrstufige Pfadbedingung zu erstellen. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).
- Sie können keine TLS-Mindestversion für Ihren privaten benutzerdefinierten Domainnamen festlegen. Für alle privaten benutzerdefinierten Domainnamen gilt die Sicherheitsrichtlinie TLS-1-2.
- Sie können eine VPC-Endpunktrichtlinie verwenden, um den Zugriff auf einen privaten benutzerdefinierten Domainnamen zu steuern. Weitere Informationen finden Sie in den Beispielen 4 und 5 unter [the section called “VPC-Endpunktrichtlinien für private Zwecke verwenden APIs”](#).
- Sie müssen eine separate Ressourcenrichtlinie für Ihre private API und für Ihren privaten benutzerdefinierten Domainnamen erstellen. Ein API-Verbraucher benötigt eine Zugriffsgenehmigung von der Ressourcenrichtlinie für private benutzerdefinierte Domainnamen, der privaten API-Ressourcenrichtlinie und allen VPC-Endpunktrichtlinien oder Genehmigungen für die private API, um einen privaten benutzerdefinierten Domainnamen aufrufen zu können.

Überlegungen zur Verwendung privater benutzerdefinierter Domainnamen mit anderen API-Gateway-Ressourcen

Die folgenden Überlegungen können Ihre Verwendung von benutzerdefinierten privaten Domainnamen mit anderen API-Gateway-Ressourcen beeinflussen:

- Es ist nicht möglich, Datenverkehr von einem privaten benutzerdefinierten Domainnamen an eine öffentliche API zu senden.
- Wenn eine private API einem privaten benutzerdefinierten Domainnamen zugeordnet ist, können Sie den Endpunkttyp der API nicht ändern.
- Sie können einen öffentlichen benutzerdefinierten Domainnamen nicht zu einem privaten benutzerdefinierten Domainnamen migrieren.
- Wenn Sie über einen VPC-Endpunkt verfügen, den Sie für den Zugriff auf einen öffentlichen benutzerdefinierten Domainnamen verwenden, können Sie diesen nicht für die Erstellung einer Domainnamenzugriffszuweisung zu einem privaten benutzerdefinierten Domainnamen verwenden.

Unterschiede zwischen privaten und öffentlichen benutzerdefinierten Domainnamen

Im Folgenden werden die Unterschiede zwischen privaten und öffentlichen benutzerdefinierten Domainnamen beschrieben:

- Private benutzerdefinierte Domainnamen müssen nicht kontenübergreifend eindeutig sein.
- Ein privater Domainname hat eine Domainnamen-ID. Diese Kennung identifiziert einen privaten benutzerdefinierten Domainnamen eindeutig und wird nicht für öffentliche benutzerdefinierte Domainnamen generiert.
- Wenn Sie den verwenden AWS CLI , um Ihren privaten benutzerdefinierten Domainnamen zu aktualisieren oder zu löschen, müssen Sie die Domainnamen-ID angeben. Wenn Sie einen privaten benutzerdefinierten Domainnamen `example.com` und einen öffentlichen benutzerdefinierten Domainnamen `example.com` haben und die Domainnamen-ID nicht angeben, wird API Gateway Ihren öffentlichen benutzerdefinierten Domainnamen ändern oder löschen.

Nächste Schritte für benutzerdefinierte Domainnamen für private APIs

Informationen zu den Aufgaben eines API-Anbieters und eines API-Verbrauchers finden Sie unter [the section called “API-Anbieter und API-Verbraucher”](#).

Anweisungen zum Erstellen eines privaten benutzerdefinierten Domainnamens, den Sie selbst aufrufen können AWS-Konto, finden Sie unter [the section called “Tutorial: Erstellen Sie einen benutzerdefinierten Domainnamen für private Zwecke und rufen Sie ihn auf APIs”](#).

Anweisungen zur Bereitstellung eines weiteren AWS-Konto Zugriffs auf Ihren privaten benutzerdefinierten Domainnamen finden Sie unter [the section called “API-Anbieter: Teilen Sie Ihren](#)

[privaten benutzerdefinierten Domainnamen mit AWS RAM](#)". Anweisungen zum Verknüpfen Ihres VPC-Endpunkts mit einem privaten benutzerdefinierten Domainnamen in einem anderen AWS-Konto finden Sie unter [the section called "API-Verbraucher: Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen"](#)

Aufgaben von API-Anbietern und API-Nutzern für benutzerdefinierte Domainnamen für private APIs

Wenn Sie einen privaten benutzerdefinierten Domainnamen erstellen, sind Sie ein API-Anbieter. Wenn Sie einen privaten benutzerdefinierten Domainnamen aufrufen, sind Sie ein API-Verbraucher. Sie können einen privaten benutzerdefinierten Domainnamen von Ihrem eigenen AWS-Konto oder einem anderen verwenden AWS-Konto.

Im folgenden Abschnitt werden die Aufgaben erläutert, die der API-Anbieter und der API-Verbraucher ausführen müssen, um einen privaten benutzerdefinierten Domainnamen verwenden zu können. Wenn Sie einen eigenen privaten benutzerdefinierten Domainnamen aufrufen möchten AWS-Konto, sind Sie sowohl der API-Anbieter als auch der API-Verbraucher. Wenn Sie eine private benutzerdefinierte Domain in einer anderen aufrufen möchten AWS-Konto, erledigt das je nach Vertrauensverhältnis zwischen dem API-Anbieter und dem API-Nutzer AWS Organizations AWS RAM möglicherweise einige Aufgaben für Sie.

Aufgaben eines API-Anbieters

API-Anbieter erstellen private Domainnamen APIs und ordnen sie benutzerdefinierten Domainnamen zu.

API-Anbieter verwalten zwei Ressourcenrichtlinien zum Schutz ihrer privaten benutzerdefinierten Domainnamen. Die erste Richtlinie bezieht sich auf den `execute-api`-Service und steuert, welche VPC-Endpunkte Ihren privaten benutzerdefinierten Domainnamen aufrufen können. In der Konfiguration des privaten benutzerdefinierten Domainnamens wird dies als `policy` bezeichnet.

Die zweite Richtlinie bezieht sich auf den Amazon API Gateway Management Service und steuert, welche VPC-Endpunkte in anderen eine Zuordnung zum Domainnamen-Zugriff mit Ihrem privaten benutzerdefinierten Domainnamen bilden AWS-Konten können. Ein VPC-Endpunkt muss eine Domainnamenzugriffszuweisung mit einem privaten benutzerdefinierten Domainnamen bilden, um ihn aufzurufen. In der Konfiguration des privaten benutzerdefinierten Domainnamens ist dies `managementPolicy`. Sie können unser API Gateway verwenden AWS RAM, um diese Richtlinie zu aktualisieren. Wenn Sie nicht vorhaben, VPC-Endpunkten in anderen Ländern AWS-Konten zu erlauben, Ihren benutzerdefinierten Domainnamen aufzurufen, bearbeiten Sie den nicht. `managementPolicy`

Wenn Sie ein API-Anbieter sind, müssen Sie Folgendes tun:

1. Erstellen einer privaten API
2. Aktualisieren Sie die `policy` Ihrer privaten API, um Ihrem VPC-Endpunkt Zugriff auf Ihre private API zu gewähren.
3. Einen privaten benutzerdefinierten Domainnamen erstellen.
4. Aktualisieren Sie die `policy` Ihres privaten benutzerdefinierten Domainnamens, um Ihrem VPC-Endpunkt Zugriff auf Ihren privaten benutzerdefinierten Domainnamen zu gewähren.
5. Erstellen Sie eine Basispfadzuweisung oder eine Routing-Regel, um Datenverkehr von Ihrer privaten API an Ihren privaten benutzerdefinierten Domainnamen zu senden. Weitere Informationen finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

Wenn Sie API-Nutzern in anderen Ländern den Zugriff auf Ihren privaten benutzerdefinierten Domainnamen ermöglichen AWS-Konten möchten, gehen Sie wie folgt vor:

1. Aktualisieren Sie die `managementPolicy` Ihres privaten benutzerdefinierten Domainnamens, damit API-Verbraucher in anderen Konten ihre VPC-Endpunkte mit Ihrem privaten benutzerdefinierten Domainnamen verknüpfen können. Gehen Sie hierzu wie folgt vor:

AWS RAM

Wenn AWS RAM sich der API-Anbieter und der API-Nutzer in derselben Organisation befinden, die sie verwendet AWS Organizations, wird die gemeinsame Nutzung der Ressourcen zwischen Anbieter und Verbraucher automatisch akzeptiert. Andernfalls sollten Sie warten, bis der API-Verbraucher die gemeinsame Nutzung der Ressource akzeptiert. Wir empfehlen Ihnen, Ihren privaten benutzerdefinierten Domainnamen mit anderen AWS RAM zu teilen.

API Gateway

Mit API Gateway AWS CLI wird nur der unterstützt. Sie müssen Ihren privaten benutzerdefinierten Domainnamen mithilfe eines Patch-Vorgangs aktualisieren und Ihr eigenes Richtliniendokument für die `managementPolicy` bereitstellen.

2. Aktualisieren Sie Ihren privaten benutzerdefinierten Domainnamen und alle ihm APIs zugewiesenen privaten Domainnamen, um Zugriff auf den VPC-Endpunkt des API-Verbrauchers zu gewähren. `policy`

Anweisungen dazu, wie Sie Ihre API einer anderen Person zur Verfügung stellen können AWS-Konto, finden Sie unter. [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen mit AWS RAM”](#)

Aufgaben eines API-Verbrauchers

API-Verbraucher verknüpfen ihre VPC-Endpunkte mit einem Domainnamen-ARN, um einen privaten benutzerdefinierten Domainnamen aufzurufen. API-Verbraucher müssen keine API-Gateway-API erstellen.

Wenn Sie ein API-Verbraucher sind, gehen Sie wie folgt vor:

1. Erstellen Sie einen VPC-Endpunkt mit aktiviertem privaten DNS in Amazon VPC.
2. (Optional — falls AWS RAM verwendet) Akzeptieren Sie AWS RAM innerhalb von 12 Stunden nach der gemeinsamen Nutzung der Ressource eine private benutzerdefinierte Domain. Wenn Sie und der API-Anbieter derselben Organisation angehören, wird die gemeinsame Nutzung der Ressource automatisch akzeptiert.
3. Rufen den privaten benutzerdefinierten Domainnamen-ARN ab. Da die URL des privaten benutzerdefinierten Domainnamens nicht eindeutig ist, verwenden Sie den privaten benutzerdefinierten Domainnamen-ARN, um die Domainnamenzugriffszuweisung zwischen Ihrem VPC-Endpunkt und dem privaten benutzerdefinierten Domainnamen zu bilden. Sie können verwenden AWS RAM , um den privaten benutzerdefinierten Domainnamen ARN abzurufen.
4. Ordnen Sie den privaten benutzerdefinierten Domain-ARN Ihrem VPC-Endpunkt in API Gateway zu. Dadurch wird eine sichere Verbindung zwischen Ihrem VPC-Endpunkt und dem privaten benutzerdefinierten Domainnamen hergestellt. Der Datenverkehr verlässt das Amazon-Netzwerk nicht.
5. Warten Sie, bis der API-Anbieter Ihrem VPC-Endpunkt Zugriff auf den privaten benutzerdefinierten Domainnamen und alle privaten, die dem privaten benutzerdefinierten Domainnamen APIs zugeordnet sind, gewährt. Wenn Sie sowohl der API-Anbieter als auch der API-Verbraucher sind, gewähren Sie Ihrem eigenen VPC-Endpunkt den Zugriff.
6. Erstellen Sie eine private gehostete Route 53-Zone und einen Route 53-Datensatz, um den privaten benutzerdefinierten Domainnamen in Route 53 aufzulösen.

Anweisungen zur Nutzung einer API in einer anderen finden Sie AWS-Konto unter. [the section called “API-Verbraucher: Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen”](#)

Tutorial: Erstellen Sie einen benutzerdefinierten Domainnamen für private Zwecke und rufen Sie ihn auf APIs

In diesem Tutorial erstellen Sie einen privaten benutzerdefinierten Domainnamen, den Sie in Ihrem eigenen Konto in einer VPC aufrufen können. Hierzu müssen Sie der API-Anbieter und der API-Verbraucher sein. Sie benötigen eine vorhandene private API und einen VPC-Endpunkt, um dieses Tutorial abzuschließen. Wenn Sie einen VPC-Endpunkt haben, den Sie für den Zugriff auf einen öffentlichen benutzerdefinierten Domainnamen verwenden, verwenden Sie diesen nicht für dieses Tutorial oder zum Erstellen von Zugriffszuweisungen für Domainnamen.

Schritt 1: Erstellen Sie einen privaten benutzerdefinierten Domainnamen

Sie erstellen Ihren privaten benutzerdefinierten Domainnamen, indem Sie den Domainnamen, das ACM-Zertifikat und die Richtlinie für den `execute-api`-Service angeben, um zu steuern, welche VPC-Endpunkte ihn aufrufen können.

AWS-Managementkonsole

So erstellen Sie einen privaten benutzerdefinierten Domainnamen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Klicken Sie auf Domainnamen hinzufügen.
4. Geben Sie unter Domänenname einen Domainnamen ein.

Ihr ACM-Zertifikat muss diesen Domainnamen abdecken, aber der Domainname muss nicht eindeutig sein.

5. Wählen Sie Privat aus.
6. Wählen Sie unter Routing-Modus die Option Nur API-Zuweisungen aus.
7. Wählen Sie unter ACM-Zertifikat ein Zertifikat aus.
8. Klicken Sie auf Domainnamen hinzufügen.

API Gateway stellt einen Domainnamen mit der Einstellung deny für alle Ressourcenrichtlinien bereit. Dies ist die Ressourcenrichtlinie für den `execute-api`-Dienst. Sie müssen diese Ressourcenrichtlinie aktualisieren, um Zugriff auf Ihre VPC-Endpunkte zu gewähren, damit Ihr privater benutzerdefinierter Domainname aufgerufen werden kann.

So aktualisieren Sie Ihre Ressourcenrichtlinie

1. Wählen Sie die Registerkarte Ressourcenrichtlinie aus und klicken Sie dann auf Ressourcenrichtlinie bearbeiten.
2. Geben Sie die folgende Ressourcenrichtlinie in den Code-Editor ein. Ersetzen Sie den VPC-Endpoint `vpce-abcd1234efg` durch Ihre eigene VPC-Endpoint-ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpce": "vpce-abcd1234"
        }
      }
    }
  ]
}
```

3. Wählen Sie Änderungen speichern aus.

AWS CLI

Wenn Sie mit dem einen privaten benutzerdefinierten Domänennamen erstellen AWS CLI, geben Sie eine Ressourcenrichtlinie für den `execute-api` Dienst an, um VPC-Endpunkten Zugriff zu gewähren, damit sie Ihren privaten benutzerdefinierten Domänennamen aufrufen können.

Verwenden Sie dazu den Parameter. `--policy file://policy.json` Sie können diese Richtlinie später ändern.

In diesem Beispiel fügen Sie die folgende Ressourcenrichtlinie als `policy` hinzu, indem Sie Parameter aus einer Datei laden. Kopieren und speichern Sie diese Datei als `policy.json`. Diese Richtlinie erlaubt nur eingehenden Datenverkehr zu einem privaten benutzerdefinierten Domainnamen vom VPC-Endpunkt `vpce-abcd1234efg` aus:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpce": "vpce-abcd1234"
        }
      }
    }
  ]
}
```

Der folgende [create-domain-name](#) Befehl erstellt einen privaten benutzerdefinierten Domainnamen:

```
aws apigateway create-domain-name \
  --domain-name 'private.example.com' \
```

```
--certificate-arn 'arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef' \
--security-policy 'TLS_1_2' \
--endpoint-configuration '{"types":["PRIVATE"]}' \
--policy file://policy.json
```

Die Ausgabe sieht wie folgt aus:

```
{
  "domainName": "private.example.com",
  "domainNameId": "abcd1234",
  "domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
  "certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
  "certificateUploadDate": "2024-09-10T10:31:20-07:00",
  "endpointConfiguration": {
    "types": [
      "PRIVATE"
    ]
  },
  "domainNameStatus": "AVAILABLE",
  "securityPolicy": "TLS_1_2",
  "routingMode" : "API_MAPPING_ONLY",
  "policy": "...
}
```

Schritt 2: Erstellen Sie eine Basispfadzuweisung, um Ihre private API Ihrem privaten benutzerdefinierten Domainnamen zuzuweisen

Nachdem Sie Ihren privaten benutzerdefinierten Domainnamen erstellt haben, ordnen Sie ihm eine private API zu. Eine Basispfadzuweisung macht eine API mithilfe der Kombination aus benutzerdefiniertem Domainnamen und dem zugehörigen Basispfad zugänglich. Wir empfehlen, dass Sie einen einzelnen privaten benutzerdefinierten Domainnamen als Hostnamen für mehrere private APIs Domains verwenden.

Alle API-Anbieter müssen eine Basispfadzuweisung erstellen, auch wenn Sie nicht vorhaben, Ihre eigene API aufzurufen. Sie müssen auch VPC-Endpunkten Zugriff gewähren, um alle privaten, APIs die Sie Ihrem privaten benutzerdefinierten Domainnamen zuordnen, aufzurufen.

AWS-Managementkonsole

So erstellen Sie eine Basispfadzuweisung

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie einen privaten benutzerdefinierten Domainnamen aus.
4. Klicken Sie in der Registerkarte API-Zuweisungen auf Zuweisungen konfigurieren.
5. Wählen Sie Neue Zuordnung hinzufügen aus.
6. Geben Sie eine API, eine Phase und optional einen Pfad ein.
7. Wählen Sie Speichern.

AWS CLI

Der folgende [create-base-path-mapping](#) Befehl erstellt eine Zuordnung zwischen einer privaten API und einem privaten benutzerdefinierten Domainnamen:

```
aws apigateway create-base-path-mapping \  
  --domain-name-id abcd1234 \  
  --domain-name 'private.example.com' \  
  --rest-api-id a1b2c3 \  
  --stage prod \  
  --base-path v1
```

Die Ausgabe sieht wie folgt aus.

```
{  
  "basePath": "v1",  
  "restApiId": "a1b2c3",  
  "stage": "prod"  
}
```

Wenn Sie mehr Flexibilität bei der Weiterleitung von Datenverkehr an Ihren APIs wünschen, können Sie den Routing-Modus auf `ROUTING_RULE_ONLY` oder ändern `ROUTING_RULE_THEN_API_MAPPING` und eine Routing-Regel erstellen. Weitere Informationen

finden Sie unter [the section called “Senden Sie Traffic APIs über Ihren benutzerdefinierten Domainnamen in API Gateway an Sie”](#).

 Note

Wenn Sie möchten AWS-Konten , dass andere Ihren privaten benutzerdefinierten Domainnamen aufrufen, folgen Sie nach Abschluss dieses Tutorials den Schritten unter [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen mit AWS RAM”](#).

Schritt 3: Erstellen Sie eine Domainnamenzugriffszuweisung zwischen Ihrem benutzerdefinierten Domainnamen und einem VPC-Endpunkt

Als Nächstes erstellen Sie eine Domainnamenzugriffszuweisung zwischen Ihrem privaten benutzerdefinierten Domainnamen und Ihrem VPC-Endpunkt. Ihr VPC-Endpunkt nutzt die Domainnamenzugriffszuweisung für den Aufruf Ihres privaten benutzerdefinierten Domainnamens, solange er vom öffentlichen Internet isoliert ist.

AWS-Managementkonsole

So erstellen Sie eine Domainnamenzugriffszuweisung

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie einen privaten benutzerdefinierten Domainnamen aus.
4. Wählen Sie auf der Registerkarte Ressourcenfreigabe unter Domainnamenzugriffszuweisungen die Option Domainnamenzugriffszuweisung erstellen aus.
5. Wählen Sie unter Domainnamen-ARN Ihren Domainnamen aus.
6. Wählen Sie unter VPC-Endpunkt-ID die VPC-Endpunkt-ID aus, auf die Sie in Schritt 1 Zugriff gewährt haben.
7. Klicken Sie auf Domainnamenzugriffszuweisung.

Sie können Ihre Zuweisung für die Domainnamenzugriffszuweisung auch auf der Seite mit den Domainnamenzugriffszuweisungen in der Konsole erstellen.

AWS CLI

Der folgende `create-domain-name-access-association`-Befehl erstellt eine Domainnamenzugriffszuweisung zwischen Ihrem privaten benutzerdefinierten Domainnamen und Ihrem VPC-Endpunkt.

```
aws apigateway create-domain-name-access-association \
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
  --access-association-source vpce-abcd1234efg \
  --access-association-source-type VPCE \
  --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{
  "domainNameAccessAssociationARN": "arn:aws:apigateway:us-west-2:111122223333:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcesource/
vpce-abcd1234efg",
  "accessAssociationSource": "vpce-abcd1234efg",
  "accessAssociationSourceType": "VPCE",
  "domainNameARN" : "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
}
```

Nach der Erstellung Ihrer Domainnamenzugriffszuweisung dauert es etwa 15 Minuten, bis diese bereitsteht. Während Sie warten, können Sie mit den folgenden Schritten fortfahren.

Schritt 4: Erstellen sie eine Route 53-gehosteten Zone

Nachdem Sie Ihre Ressourcenrichtlinie aktualisiert und Ihren privaten benutzerdefinierten Domainnamen Ihrem VPC-Endpunkt zugewiesen haben, erstellen Sie eine private gehostete Zone in Route 53, um Ihren benutzerdefinierten Domainnamen aufzulösen. Eine gehostete Zone ist ein Container, der Informationen darüber enthält, wie Sie den Verkehr für eine Domain innerhalb einer oder mehrerer Domains weiterleiten möchten, VPCs ohne Ihre Ressourcen dem Internet auszusetzen. Weitere Informationen finden Sie unter [Arbeiten mit privat gehosteten Zonen](#).

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung [von finden Sie unter Creating a private hosted zone](#) im Amazon Route 53 Developer Guide.

Geben Sie unter Name den Namen Ihres privaten benutzerdefinierten Domainnamens an. Verwenden Sie für VPC-ID die VPC, die den VPC-Endpunkt enthält, den Sie in den Schritten oben verwendet haben.

AWS CLI

Der folgende [create-hosted-zone](#) Befehl erstellt eine private gehostete Zone:

```
aws route53 create-hosted-zone --name private.example.com \
  --caller-reference 2014-04-01-18:47 \
  --hosted-zone-config Comment="command-line version",PrivateZone=true \
  --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

Die Ausgabe enthält die ID der gehosteten Zone. Die ID der gehosteten Zone brauchen Sie für die nachfolgenden Schritte.

Schritt 5: Erstellen eines DNS-Datensatzes mithilfe von Route 53

Nachdem Sie die gehostete Zone erstellt haben, erstellen Sie einen Datensatz, um Ihren privaten benutzerdefinierten Domainnamen aufzulösen. Hierzu verwenden Sie die ID der gehosteten Zone, die Sie im Schritt oben erstellt haben. In diesem Beispiel erstellen Sie einen A-Datensatztyp. Wenn Sie IPv6 für Ihren VPC-Endpunkt verwenden, erstellen Sie einen AAAA-Eintragstyp. Wenn Sie Dualstack für Ihren VPC-Endpunkt verwenden, erstellen Sie sowohl einen AAAA- als auch einen A-Datensatztyp.

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung von finden Sie unter [Weiterleiten von Datenverkehr an eine Amazon API Gateway Gateway-API mithilfe Ihres Domainnamens](#).

Verwenden Sie die Funktion Schnellerstellung und aktivieren Sie Alias. Für Endpunkt nehmen Sie den DNS-Namen des VPC-Endpunkts.

AWS CLI

Wenn Sie Ihre DNS-Datensätze so konfigurieren möchten, dass der private benutzerdefinierte Domainname dem Hostnamen der vorgegebenen gehosteten Zonen-ID zugewiesen wird, erstellen Sie eine JSON-Datei, die die Konfiguration für die Einrichtung eines DNS-Datensatzes für den privaten Domainnamen enthält.

Im Folgenden `setup-dns-record.json` wird gezeigt, wie Sie einen DNSA-Datensatz erstellen, um einen privaten benutzerdefinierten Domainnamen seinem privaten Hostnamen zuzuweisen.

Sie geben den `DNSName` Ihrer VPC-DNS-ID und die `Hosting-Zonen-ID` an, die Sie oben erstellt haben.

```
{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z20JLYMU09EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Der folgende [change-resource-record-sets](#) Befehl erstellt einen DNS-Eintrag für Ihren privaten benutzerdefinierten Domainnamen:

```
aws route53 change-resource-record-sets \
  --hosted-zone-id ZABCDEFGHI1234 \
  --change-batch file://path/to/your/setup-dns-record.json
```

Ersetzen Sie die `hosted-zone-id` durch die Route-53-Hosted-Zone-ID des DNS-Datensatzes, die in Ihrem Konto eingerichtet wurde. Der `change-batch`-Parameterwert verweist auf eine JSON-Datei.

Wenn Sie nicht vorhaben, Ihren eigenen privaten benutzerdefinierten Domainnamen aufzurufen, können Sie diese Ressourcen löschen, nachdem Sie bestätigt haben, dass Ihr privater benutzerdefinierter Domainname funktioniert.

Schritt 6: Aufrufen Ihres privaten benutzerdefinierten Domainnamens

Sie können Ihren privaten benutzerdefinierten Domainnamen jetzt in Ihrem eigenen AWS-Konto aufrufen. Verwenden Sie in Ihrer VPC den folgenden `curl`-Befehl, um auf Ihren privaten benutzerdefinierten Domainnamen zuzugreifen.

```
curl https://private.example.com/v1
```

Weitere Informationen zu anderen Möglichkeiten Ihre private API aufzurufen, finden Sie unter [the section called “Aufrufen einer privaten API mithilfe eines benutzerdefinierten Domainnamens”](#).

Schritt 7: Bereinigen

Löschen Sie die Verknüpfung zwischen Ihrem VPC-Endpunkt und Ihrem privaten benutzerdefinierten Domainnamen, um unnötige Kosten zu vermeiden, und löschen Sie anschließend Ihren privaten benutzerdefinierten Domainnamen.

AWS-Managementkonsole

So löschen Sie die Domainnamenzugriffszuweisung

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Domainnamenzugriffszuweisungen.
3. Wählen Sie Ihre Domainnamenzugriffszuweisung aus und klicken Sie dann auf Löschen.
4. Bestätigen Sie Ihr Auswahl und klicken Sie auf Löschen.

Nachdem Sie Ihre Domainnamenzugriffszuweisung gelöscht haben, können Sie nun Ihren privaten benutzerdefinierten Domainnamen löschen.

So löschen Sie Ihren privaten benutzerdefinierten Domainnamen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie Ihren privaten benutzerdefinierten Domainnamen aus.
4. Klicken Sie auf Löschen.
5. Bestätigen Sie Ihr Auswahl und klicken Sie auf Löschen.

Falls nötig, können Sie auch Ihren VPC-Endpunkt löschen. Weitere Informationen finden Sie unter [Löschen eines Schnittstellenendpunkts](#).

AWS CLI

So räumen Sie auf

1. Der Befehl `delete-access-association` löscht die Domainnamenzugriffszuweisung:

```
aws apigateway delete-domain-name-access-association \  
  --domain-name-access-association-arn 'arn:aws:apigateway:us-  
west-2:111122223333:/domainnameaccessassociations/domainname/private.example.com  
+abcd1234/vpcsource/vpce-abcd1234efg' \  
  --region us-west-2
```

2. Der Befehl `delete-domain-name` löscht Ihren privaten benutzerdefinierten Domainnamen. Mit diesem Befehl werden auch alle Basispfadzuweisungen entfernt.

```
aws apigateway delete-domain-name \  
  --domain-name test.private.com \  
  --domain-name-id abcd1234
```

Falls nötig, können Sie auch Ihren VPC-Endpunkt löschen. Weitere Informationen finden Sie unter [Löschen eines Schnittstellenendpunkts](#).

Best Practices

Wir empfehlen die Verwendung der folgenden bewährten Methoden zur Erstellung Ihres privaten benutzerdefinierten Domainnamens:

- Verwenden Sie grundlegende Pfadzuordnungs- oder Routing-Regeln, um Traffic von einem privaten benutzerdefinierten Domainnamen an mehrere private zu senden. APIs
- Wenn ein VPC-Endpunkt keinen Zugriff mehr auf einen privaten benutzerdefinierten Domainnamen benötigt, löschen Sie die Zuweisung. Entfernen Sie außerdem den VPC-Endpunkt aus der `policy` für den `execute-api`-Service für die private benutzerdefinierte Domain.
- Konfigurieren Sie mindestens zwei Verfügbarkeitszonen für jeden VPC-Endpunkt.
- Deaktivieren des Standardendpunkts Wir empfehlen, den Standardendpunkt zu deaktivieren, damit Ihre API-Verbraucher Ihre API nur über den benutzerdefinierten Domainnamen aufrufen können. Weitere Informationen finden Sie unter [the section called “Deaktivieren des Standardendpunkts”](#).
- Wir empfehlen, bei der Einrichtung Ihres privaten benutzerdefinierten Domainnamens eine private gehostete Route 53-Zone und einen Datensatz des Typs A bereitzustellen. Wenn Sie nicht

vorhaben, Ihren eigenen privaten benutzerdefinierten Domainnamen aufzurufen, können Sie diese Ressourcen später löschen.

Arbeiten mit kontoübergreifenden privaten benutzerdefinierten Domainnamen

In diesem Abschnitt wird erläutert, wie Sie mit kontoübergreifenden privaten benutzerdefinierten Domainnamen arbeiten. Sie können einem anderen einen privaten benutzerdefinierten Domainnamen zur Verfügung stellen AWS-Konto und einen anderen verwenden AWS-Konto , um einen privaten benutzerdefinierten Domainnamen aufzurufen.

Sie können Ihren privaten benutzerdefinierten Domainnamen AWS-Konto mithilfe AWS Resource Access Manager unseres API Gateway an andere weitergeben. AWS Resource Access Manager (AWS RAM) hilft Ihnen dabei, Ihre Ressourcen innerhalb AWS-Konten und innerhalb Ihrer Organisation oder Organisationseinheiten sicher zu teilen (OUs). Weitere Informationen finden Sie unter [Was ist AWS Resource Access Manager](#).

Anweisungen dazu, wie Sie einen privaten benutzerdefinierten Domainnamen mit einem anderen Benutzer teilen AWS-Konto können AWS RAM, finden Sie unter [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen mit AWS RAM”](#).

Anweisungen dazu, wie Sie AWS-Konto mithilfe von API Gateway einen privaten benutzerdefinierten Domainnamen mit einem anderen teilen können, finden Sie unter [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen über das API Gateway AWS CLI”](#).

Anweisungen dazu, wie Sie einen privaten benutzerdefinierten Domainnamen in einem anderen verwenden können AWS-Konto, finden Sie unter [the section called “API-Verbraucher: Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen”](#).

Bewährte Methoden für die Arbeit mit kontoübergreifenden privaten benutzerdefinierten Domainnamen

Wir empfehlen die folgenden bewährten Methoden für die Arbeit mit kontoübergreifenden privaten benutzerdefinierten Domainnamen:

- Verwenden Sie AWS RAM diese Option, um Ihre privaten benutzerdefinierten Domainnamen zu teilen. Wenn Sie es verwenden AWS RAM, können Sie den Betriebsaufwand reduzieren und müssen keinen Service managementPolicy für das Amazon API Gateway Management einrichten.

- Verwenden Sie den `resource-owner`-Parameter zum Auflisten Ihrer privaten benutzerdefinierten Domainnamen oder Domainnamenzugriffszuweisungen. Verwenden Sie den `resource-owner`-Parameter, wenn Sie nur die Ressourcen auflisten möchten, die Ihnen oder anderen AWS-Konten gehören.

Im folgenden Beispiel sehen Sie, wie Sie alle Domainnamenzugriffszuweisung abrufen, die Sie besitzen:

```
aws apigateway get-domain-name-access-associations --resource-owner SELF
```

Verwenden Sie `--resource-owner OTHER_ACCOUNTS`, um alle Domainnamenzugriffszuweisungen aufzulisten, die andere Konten mit Ihrem privaten benutzerdefinierten Domainnamen gebildet haben.

API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen mit AWS RAM

Sie können API-Nutzern in anderen Fällen AWS-Konten Zugriff auf Ihren privaten benutzerdefinierten Domainnamen gewähren. In diesem Abschnitt erfahren Sie, wie Sie Ihren privaten benutzerdefinierten Domainnamen mit anderen teilen AWS RAM und wie Sie den Zugriff auf Ihren privaten benutzerdefinierten Domainnamen kontrollieren können.

Überlegungen zur Weitergabe Ihres privaten benutzerdefinierten Domainnamens

Die folgenden Überlegungen können sich darauf auswirken, wie Sie mithilfe von Zugriff auf Ihren privaten benutzerdefinierten Domainnamen mithilfe von AWS RAM gewähren. Informationen dazu, wie Sie Ihren privaten benutzerdefinierten Domainnamen teilen können, ohne ihn zu verwenden AWS RAM, finden Sie unter [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen über das API Gateway AWS CLI”](#).

- Private benutzerdefinierte Domainnamen werden auf der gleichen AWS-Region Ebene geteilt. Der private benutzerdefinierte Domainname und der VPC-Endpunkt müssen sich in derselben AWS-Region befinden.
- Sie können eine einzelne Ressourcenfreigabe für mehrere Prinzipale verwenden und weitere Prinzipale hinzufügen, nachdem Sie die Ressourcenfreigabe erstellt haben. Wir empfehlen, dass Sie Ihre Ressourcenfreigabe nach Möglichkeit wiederverwenden.
- Sie müssen dem VPC-Endpunkt des API-Verbrauchers immer Zugriff gewähren, um Ihren privaten benutzerdefinierten Domainnamen und alle ihm APIs zugewiesenen privaten Domainnamen aufzurufen.

- Wenn sich der API-Nutzer und der API-Anbieter in derselben Organisation befinden, die sie verwendet AWS Organizations, wird die gemeinsame Nutzung der Ressource automatisch akzeptiert. Sie müssen jedoch trotzdem mithilfe von AWS RAM eine Ressourcenfreigabe erstellen.
- Wenn sich der API-Nutzer und der API-Anbieter in derselben Organisation befinden AWS Organizations und die gemeinsame Nutzung von Ressourcen innerhalb Ihrer Organisation aktiviert ist, wird allen Prinzipalen in der Organisation, für die Sie die gemeinsame Nutzung nutzen, automatisch Zugriff auf die Ressourcenfreigaben gewährt. Es ist keine Einladung erforderlich und Sie können die gemeinsame Ressourcennutzung überspringen.
- Wenn der API-Verbraucher die gemeinsame Ressourcennutzung nicht innerhalb von 12 Stunden akzeptiert, muss der API-Anbieter die Ressource erneut teilen.
- Nachdem Sie die Ressourcenfreigabe erstellt haben, AWS RAM aktualisiert sie den Service managementPolicy für den Amazon API Gateway Management Service für Ihren privaten benutzerdefinierten Domainnamen, um den Zugriff auf Prinzipale ohne expliziten allow Zugriff zu verhindern. Weitere Informationen finden Sie unter [Ermitteln, ob eine Anforderung innerhalb eines Kontos zugelassen oder verweigert wird](#) im IAM-Benutzerhandbuch.

Die aktualisierte managementPolicy wird wie folgt aussehen:

JSON

```
{
  "Version": "2012-10-17",
  "Id": "abcd1234-1234-abcd-abcd-1234abcdefg",
  "Statement": [
    {
      "Sid":
      "APIGatewayPrivateDomainNameManagementPolicyDefaultPermission-org",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "apigateway:CreateAccessAssociation",
      "Resource": "arn:aws:apigateway:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "o-1234abcd"
        },
        "StringNotEquals": {
          "aws:PrincipalAccount": "111122223333"
        }
      }
    }
  ]
}
```



```
}  
]  
}
```

AWS RAM hat verhindert, dass Prinzipale ohne expliziten `allow` Zugriff Zugriffszuordnungen mit Ihrem privaten benutzerdefinierten Domainnamen erstellen können, indem Folgendes hinzugefügt wurde:

```
"StringNotEquals": {  
  "aws:PrincipalAccount": "111122223333"  
}
```

Sie können weiterhin den Prinzipal in der Person verwenden AWS-Konto, die den privaten benutzerdefinierten Domainnamen erstellt hat, um Zugriffszuordnungen für Domainnamen zu erstellen.

Anderen Konten erlauben, Domainnamen Zugriffszuweisungen mit Ihrem privaten benutzerdefinierten Domainnamen zu erstellen

Zunächst gewähren Sie einer anderen Person Zugriff, AWS-Konto um Zugriffszuordnungen für Domainnamen mit Ihrem privaten benutzerdefinierten Domainnamen zu erstellen.

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung [von finden Sie unter Creating a Resource Share AWS RAM im AWS RAM](#) Benutzerhandbuch.

Wählen Sie unter Ressourcentyp auswählen die Option Private benutzerdefinierte API-Gateway-Domains aus.

AWS CLI

Im Folgenden [create-resource-share](#) wird eine Ressourcenfreigabe für Ihren privaten benutzerdefinierten Domainnamen erstellt. Es kann einige Minuten dauern, bis die Ressourcen- und Prinzipal-Zuordnungen abgeschlossen ist. Geben Sie für Prinzipale eine Konto-ID oder eine Organisations-ID an (z. B. `arn:aws:organizations::123456789012:organization/o-1234abcd`). Sie können mehrere Prinzipale für Ihre Ressourcenfreigabe angeben.

```
aws ram create-resource-share \  
  --region us-west-2 \  
  --principals arn:aws:iam::123456789012:user/JohnDoe
```

```
--name privateCustomDomain-resource-share \  
--permission-arns arn:aws:ram::aws:permission/  
APIGatewayPrivateDomainNameManagementPolicyDefaultPermission \  
--resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/  
private.example.com+abcd1234 \  
--principals 222222222222
```

Nachdem Sie einem anderen Zugriff gewährt haben AWS-Konto, müssen API-Verbraucher in diesem Konto eine Zuordnung für den Domainnamen-Zugriff zwischen ihrem VPC-Endpunkt und Ihrem privaten benutzerdefinierten Domainnamen erstellen. Diese Domainnamen-zugriffszuweisung können Sie nicht für andere erstellen. Weitere Informationen finden Sie unter [the section called “Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen”](#).

Anderen Konten Aufrufberechtigungen für Ihren privaten benutzerdefinierten Domainnamen gewähren

Als Nächstes gewähren Sie dem VPC-Endpunkt des API-Verbrauchers Zugriff, um Ihren privaten benutzerdefinierten Domainnamen und alle ihm APIs zugewiesenen privaten Domainnamen aufzurufen.

AWS-Managementkonsole

VPC-Endpunkten in anderen Konten Zugriff auf Ihren privaten benutzerdefinierten Domainnamen gewähren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie den privaten benutzerdefinierten Domainnamen, den Sie mit anderen geteilt haben. AWS-Konten
4. Klicken Sie auf der Registerkarte Ressourcenrichtlinie auf Ressourcenrichtlinie bearbeiten.
5. Fügen Sie die VPC-Endpunkt-ID des API-Verbrauchers zu Ihrer Ressourcenrichtlinie hinzu.

Sie finden die VPC-Endpunkt-ID des API-Verbrauchers im Abschnitt Domainnamen-zugriffszuweisungen auf der Registerkarte Ressourcenfreigabe auf der Seite mit den Domaindetails Ihres privaten benutzerdefinierten Domainnamens.

6. Klicken Sie auf Änderungen speichern.

AWS CLI

Die folgende policy für den `execute-api`-Service ermöglicht eingehenden Datenverkehr zu einem privaten benutzerdefinierten Domainnamen vom VPC-Endpoint `vpce-abcd1234efg` und `vpce-xyz000abc`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpce": [
            "vpce-abcd1234",
            "vpce-xyz0000"
          ]
        }
      }
    }
  ]
}
```

Der folgende [update-domain-name](#) Befehl verwendet einen Patch-Vorgang, um den policy für einen privaten benutzerdefinierten Domainnamen zu aktualisieren:

```
aws apigateway update-domain-name
  --domain-name private.example.com \
```

```
--domain-name-id abcd1234 \
--patch-operations op=replace,path=/policy,value='{\"Version\": \"2012-10-17\",
  \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\":
    \"execute-api:Invoke\", \"Resource\": [\"execute-api:/*\"]}, {\"Effect\": \"Deny\",
    \"Principal\": \"*\", \"Action\": \"execute-api:Invoke\", \"Resource\": [\"execute-
    api:/*\"], \"Condition\": {\"StringNotEquals\": [\"vpce-abcd1234efg\", \"vpce-xyz000abc
    \"]}]}'\"
```

API-Anbieter: Beenden Sie die gemeinsame Nutzung eines privaten benutzerdefinierten Domainnamens mit AWS RAM

Um die gemeinsame Nutzung Ihres privaten benutzerdefinierten Domainnamens zu beenden, verhindern Sie zunächst, dass der API-Verbraucher weitere Domainnamenzugriffszuweisungen erstellt, indem Sie die gemeinsame Nutzung der Ressource aufheben. Anschließend weisen Sie die Domainnamenzugriffszuweisung zurück und entfernen den VPC-Endpunkt des API-Verbrauchers aus Ihrer `policy` für den `execute-api`-Service. Der API-Verbraucher kann dann seine Domainnamenzugriffszuweisung löschen.

Gemeinsame Nutzung Ihres privaten benutzerdefinierten Domainnamens beenden

Zunächst beenden Sie die Nutzung von Resource Share AWS RAM.

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung von finden Sie unter [Aktualisieren einer Ressourcenfreigabe in AWS RAM](#).

AWS CLI

Im Folgenden [disassociate-resource-share](#) wird die Zuordnung einer Ressourcenfreigabe für Ihren privaten benutzerdefinierten Domainnamen aufgehoben.

```
aws ram disassociate-resource-share \
  --region us-west-2 \
  --resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/
  private.example.com+abcd1234 \
  --principals 222222222222
```

Domainnamenzugriffszuweisung zurückweisen

Nachdem Sie die gemeinsame Nutzung Ihrer Ressource beendet haben AWS RAM, lehnen Sie die Zuordnung des Domainnamen-Zugriffs zwischen einem VPC-Endpunkt in einem anderen Konto und Ihrem privaten benutzerdefinierten Domainnamen ab.

Note

In Ihrem eigenen Konto können Sie eine Domainnamenzugriffszuweisung nicht zurückweisen. Löschen Sie die Domainnamenzugriffszuweisung, um die gemeinsame Nutzung der Ressource zu beenden. Weitere Informationen finden Sie unter [Löschen einer Domainnamenzugriffszuweisung](#).

Wenn Sie eine Domainnamenzugriffszuweisung zu einem VPC-Endpunkt zurückweisen und ein API-Verbraucher versucht, Ihren privaten benutzerdefinierten Domainnamen aufzurufen, lehnt API Gateway den Aufruf ab und gibt einen 403-Statuscode zurück.

AWS-Managementkonsole

Abweisen einer Domainnamenzugriffszuweisung

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie den privaten benutzerdefinierten Domainnamen, den Sie mit anderen geteilt haben. AWS-Konten
4. Wählen Sie unter Ressourcenfreigabe die Domainnamenzugriffszuweisung aus, die Sie abweisen möchten.
5. Klicken Sie auf Zuweisung ablehnen.
6. Bestätigen Sie Ihr Auswahl und klicken Sie auf Ablehnen.

AWS CLI

Der Befehl `reject-domain-name-access-association` erstellt eine Domainnamenzugriffszuweisung zwischen dem VPC-Endpunkt und Ihrem privaten benutzerdefinierten Domainnamen:

```
aws apigateway reject-domain-name-access-association \
  --domain-name-access-association-arn arn:aws:apigateway:us-west-2:444455556666:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcsource/
vpce-abcd1234efg \
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

Einem API-Anbieter den Zugriff auf Ihren privaten benutzerdefinierten Domainnamen verweigern

Nachdem Sie die Zuweisung für die Domainnamenzugriffszuweisung abgelehnt haben, entfernen Sie den VPC-Endpunkt des API-Verbrauchers aus Ihrer `policy` für den `execute-api`-Service.

AWS-Managementkonsole

Entfernen des VPC-Endpunkts des API-Verbrauchers aus Ihrer Ressourcenrichtlinie

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Benutzerdefinierte Domainnamen.
3. Wählen Sie den privaten benutzerdefinierten Domainnamen, den Sie mit anderen geteilt haben. AWS-Konten
4. Klicken Sie in der Registerkarte Zugriffsrichtlinie auf Bearbeiten.
5. Entfernen Sie den VPC-Endpunkt aus der Richtlinie.
6. Wählen Sie Änderungen speichern aus.

AWS CLI

Der folgende `update-domain-name` Befehl verwendet einen Patch-Vorgang, um den `execute-api` Dienst `policy` für einen privaten benutzerdefinierten Domainnamen zu aktualisieren. Diese neue `policy` entfernt eine zusätzliche VPC-Endpunkt-ID, die im [the section called “Anderen Konten Aufrufberechtigungen für Ihren privaten benutzerdefinierten Domainnamen gewähren”](#) hinzugefügt wurde:

```
aws apigateway update-domain-name
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations op=replace,path=/policy,value='{"Version": "2012-10-17",
  "Statement": [{"Effect": "Allow", "Principal": "*", "Action":
```

```

{"execute-api:Invoke": [{"Resource": ["execute-api:/*"]}], {"Effect": "Deny",
"Principal": "*", "Action": "execute-api:Invoke", "Resource": ["execute-
api:/*"], "Condition": {"StringNotEquals": {"aws:SourceVpce": "vpce-abcd1234efg
"}}}}}]

```

Der API-Verbraucher sollte dann die Domainnamenzugriffszuweisung löschen. Sie können das nicht von Ihrer Seite aus tun. Weitere Informationen finden Sie unter [the section called “API-Verbraucher: Löschen Ihrer Domainnamenzugriffszuweisung zu einem privaten benutzerdefinierten Domainnamen”](#).

API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen über das API Gateway AWS CLI

Sie können einen privaten benutzerdefinierten Domainnamen mit dem API Gateway teilen AWS CLI, wir empfehlen jedoch, dies AWS RAM zu verwenden, um Ihren betrieblichen Aufwand zu reduzieren. Anweisungen dazu, wie Sie AWS RAM Ihren privaten benutzerdefinierten Domainnamen mit anderen teilen können, finden Sie unter [the section called “API-Anbieter: Teilen Sie Ihren privaten benutzerdefinierten Domainnamen mit AWS RAM”](#).

Um einen privaten benutzerdefinierten Domainnamen mithilfe des API Gateway zu teilen AWS CLI, gewähren Sie anderen AWS-Konten Zugriff, um Domainnamen-Zugriffszuordnungen zu erstellen und Ihren privaten benutzerdefinierten Domainnamen aufzurufen. Hierzu aktualisieren Sie die `managementPolicy` für den API-Gateway-Management-Dienst und die `policy` für den `execute-api`-Dienst Ihres privaten benutzerdefinierten Domainnamens. Sie müssen in der Ressourcenrichtlinie auch Zugriff auf den VPC-Endpunkt des API-Verbrauchers für alle privaten Benutzer gewähren, die Ihrem privaten benutzerdefinierten Domainnamen APIs zugeordnet sind.

Der API-Verbraucher muss trotzdem eine Domainnamenzugriffszuweisung zwischen dem eigenen VPC-Endpunkt und Ihrem privaten benutzerdefinierten Domainnamen erstellen. Das können Sie nicht von Ihrer Seite aus tun.

Zugriff auf Ihren privaten benutzerdefinierten Domainnamen gewähren

So gewähren Sie Zugriff auf Ihren privaten benutzerdefinierten Domainnamen

1. Erstellen Sie eine JSON-Datei mit dem Patch-Vorgang zur Aktualisierung der Richtlinie, um die `managementPolicy` für den API-Gateway-Management-Dienst zu aktualisieren. Im Folgenden wird die aktuelle Richtlinie durch eine Beispielryrichtlinie `patch-managementPolicy.json` ersetzt, die AWS-Konten 111122223333 und 444455556666 Zugriff

gewährt, um Domännennamen-Zugriffszuordnungen managementPolicy mit dem privaten benutzerdefinierten Domännennamen zu erstellen. `private.example.com`

```
[{
  "op": "replace",
  "path": "/managementPolicy",
  "value": "{\"Version\":\"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": {\"AWS\": [\"arn:aws:iam:111122223333:root\", \"arn:aws:iam:444455556666:root\"]}, \"Action\": \"apigateway:CreateAccessAssociation\", \"Resource\": \"arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234\"}]}"
}]
```

[update-domain-name](#) Der managementPolicy folgende Befehl aktualisiert die Verwendung von. `patch-managementPolicy.json`

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-managementPolicy.json
```

Sobald Sie den Zugriff gewährt haben, müssen Sie den API-Verbraucher informieren, dass er die Domainnamen zugriffszuweisung erstellen kann. Wenn Sie verwenden AWS RAM, AWS RAM erledigt ich diesen Schritt für Sie.

- Erstellen Sie eine JSON-Datei mit dem Patch-Vorgang zur Aktualisierung der Richtlinie, um die policy für den execute-api-Dienst zu aktualisieren. Die `patch-policy.json` ersetzt die aktuelle policy durch eine Beispielrichtlinie, die zwei VPC-Endpunkten das Aufrufen des privaten benutzerdefinierten Domainnamens `private.example.com` gewährt.

```
[{
  "op": "replace",
  "path": "/policy",
  "value": "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"execute-api:Invoke\", \"Resource\": \"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com+abcd1234\"}, {\"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"execute-api:Invoke\", \"Resource\": \"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com+abcd1234\", \"Condition\": {\"StringNotEquals\": {\"aws:SourceVpce\": [\"vpce-abcd1234\", \"vpce-xyzz0000\"]}}}]}"
}]
```



```
}]
```

Verwenden Sie den folgenden [update-domain-name](#) Befehl, um die policy Verwendung zu aktualisieren `patch-policy.json`.

```
aws apigateway update-domain-name \  
  --domain-name private.example.com \  
  --domain-name-id abcd1234 \  
  --patch-operations file://patch-policy.json
```

Zugriff auf Ihren privaten benutzerdefinierten Domainnamen verweigern

Wenn Sie die Weitergabe Ihres privaten benutzerdefinierten Domainnamens beenden möchten, müssen Sie die Domainnamenzugriffszuweisungen zwischen Ihrem privaten benutzerdefinierten Domainnamen und dem VPC-Endpunkt des API-Verbrauchers zurückweisen.

So lehnen Sie den Zugriff auf Ihren privaten benutzerdefinierten Domainnamen ab

1. Der Befehl `reject-domain-name-access-association` weist die Domainnamenzugriffszuweisung ab.

```
aws apigateway reject-domain-name-access-association \  
  --domain-name-access-association-arn arn:aws:apigateway:us-  
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com  
+abcd1234/vpcesource/vpce-abcd1234 \  
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/  
private.example.com+abcd1234
```

2. Ändern Sie die Datei `patch-managementPolicy.json`, um den Zugriff für das Konto des API-Anbieters zu entfernen, damit keine Domainnamenzugriffszuweisung mit Ihrem privaten benutzerdefinierten Domainnamen erstellt werden kann. Die Datei `patch-managementPolicy.json` entfernt ein Konto aus der `managementPolicy`:

```
[{  
  "op": "replace",  
  "path": "/managementPolicy",  
  "value": "{\"Version\":\"2012-10-17\", \"Statement\": [{\"Effect\":  
\"Allow\", \"Principal\": \"*\", \"Action\": \"apigateway:CreateAccessAssociation  
\", \"Resource\": \"arn:aws:apigateway:us-west-2:111122223333:/domainnames/  
private.example.com+abcd1234\"}]}"
```

```
}]
```

Der folgende [update-domain-name](#) Befehl aktualisiert die managementPolicy Verwendung patch-managementPolicy.json.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-managementPolicy.json
```

3. Bearbeiten Sie die Datei patch-policy.json, um den Zugriff für den VPC-Endpunkt des API-Anbieters zu entfernen und den Aufruf Ihres privaten benutzerdefinierten Domainnamens zu unterbinden. Die patch-policy.json entfernt die VPC-Endpunkt-ID aus der policy:

```
[{
  "op": "replace",
  "path": "/policy",
  "value": "{\n\"Version\": \"2012-10-17\",      \"Statement\": [{\n\"Effect\": \n\"Allow\", \"Principal\": \"*\", \"Action\": \"execute-api:Invoke\", \"Resource\": \n\"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com\n+abcd1234\", {\n\"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"execute-\napi:Invoke\", \"Resource\": \"arn:aws:execute-api:us-west-2:111122223333:/\ndomainnames/private.example.com+abcd1234\", \"Condition\": {\n\"StringNotEquals\": \n{\n\"aws:SourceVpce\": \"vpce-abcd1234\"}}}}]\n}"
}]
```

Der folgende [update-domain-name](#) Befehl aktualisiert die policy Verwendung patch-policy.json.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-policy.json
```

In diesem Verfahren verwendete Beispielrichtlinien

Im folgenden Abschnitt sehen Sie die Beispielrichtlinien, die im Verfahren oben verwendet wurden.

Die folgende Beispielrichtlinie bezieht sich auf die managementPolicy des Amazon-API-Gateway-Management-Dienstes. Diese Richtlinie gewährt AWS-Konten 111122223333 und 444455556666

Zugriff zum Erstellen von Domännennamenzugriffszuordnungen mit dem privaten benutzerdefinierten Domännennamen. `private.example.com`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "111122223333",
          "444455556666"
        ]
      },
      "Action": "apigateway:CreateAccessAssociation",
      "Resource": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+a1b2c3"
    }
  ]
}
```

Die folgende Beispielrichtlinie ist die Richtlinie für die `policy` des `execute-api`-Dienstes. Diese Richtlinie gewährt den VPC-Endpunkten `vpce-abcd1234` und `vpce-xyzz0000` Zugriff zum Aufrufen des privaten benutzerdefinierten Domainnamens.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
    },
    {
```

```
    "Effect": "Deny",
    "Principal": "*",
    "Action": "execute-api:Invoke",
    "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
    "Condition": {
        "StringNotEquals": {
            "aws:SourceVpce": [
                "vpce-abcd1234",
                "vpce-xyzz0000"
            ]
        }
    }
}
```

API-Verbraucher: Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen

Das folgende Verfahren zeigt, wie Sie einen privaten Domainnamen in einem anderen verwenden AWS-Konto. Abhängig von Ihrer Vertrauensbeziehung mit dem API-Anbieter AWS RAM kann es sein, dass er einige Aufgaben für Sie erledigt.

Wenn Sie sich in einem anderen AWS-Konto als einem privaten benutzerdefinierten Domainnamen befinden, können Sie Ihren VPC-Endpunkt nur einem privaten benutzerdefinierten Domainnamen zuordnen und ihn aufrufen. Sie können die `policy` oder andere Parameter des privaten benutzerdefinierten Domainnamens nicht anzeigen.

Voraussetzungen

Die folgenden Voraussetzungen sind erforderlich, um einen privaten benutzerdefinierten Domainnamen in einem anderen zu verwenden: AWS-Konto

- VPC und VPC-Endpunkt für den `execute-api`-Service erstellen. In Ihrem VPC muss `enableDnsHostnames` und `enableDnsSupport` auf `true` gesetzt sein.
- Wir empfehlen, mindestens zwei Verfügbarkeitszonen für jeden VPC-Endpunkt zu konfigurieren.

(Optional) Akzeptieren Sie die Ressourcenfreigabe für die private benutzerdefinierte Domain

Wenn Ihr API-Anbieter früher AWS RAM eine Ressourcenfreigabe erstellt hat, haben Sie 12 Stunden Zeit, diese zu akzeptieren. Wenn Sie derselben Organisation angehören, die den API-Anbieter verwendet AWS Organizations, wird die Freigabe automatisch akzeptiert. Wenn Sie einer Organisation angehören, in der die automatische Ressourcenfreigabe aktiviert ist, wird die Ressource automatisch für Sie freigegeben.

AWS-Managementkonsole

Informationen zur Verwendung von finden Sie im AWS-ManagementkonsoleAWS RAM Benutzerhandbuch unter [Annehmen und Ablehnen von Einladungen zur gemeinsamen Nutzung von Ressourcen](#).

AWS CLI

Verwenden Sie den folgenden [get-resource-share-invitations](#)Befehl, um alle Ressourcen zu finden, die mit Ihnen geteilt wurden:

```
aws ram get-resource-share-invitations \
  --region us-west-2
```

Verwenden Sie den resultierenden Ressourcenfreigabe-ARN, um die Einladung zur Ressourcenfreigabe anzunehmen. Der folgende [accept-resource-share-invitation](#)Befehl akzeptiert die gemeinsame Nutzung der Ressource.

```
aws ram accept-resource-share-invitation \
  --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
  --region us-west-2
```

Zuweisung Ihres VPC-Endpunkts zu einem mit Ihnen gemeinsam genutzten privaten benutzerdefinierten Domainnamen

Da private benutzerdefinierte Domainnamen nicht eindeutig sind, verknüpfen Sie Ihren VPC-Endpunkt mit dem eindeutigen benutzerdefinierten Domainnamen-ARN. Nachdem Sie Ihre Domainnamenzugriffszuweisung erstellt haben, kann es bis zu 15 Minuten dauern, bis Ihr VPC-Endpunkt Ihren privaten benutzerdefinierten Domänennamen erfolgreich aufruft. Wenn Sie über einen VPC-Endpunkt verfügen, den Sie für den Zugriff auf einen öffentlichen

benutzerdefinierten Domainnamen verwenden, sollten Sie diesen nicht zur Erstellung einer Domainnamenzugriffszuweisung verwenden.

AWS-Managementkonsole

So verknüpfen Sie Ihren VPC-Endpunkt mit einem gemeinsam genutzten privaten benutzerdefinierten Domainnamen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Klicken Sie im Hauptnavigationsbereich auf Domainnamenzugriffszuweisungen.
3. Klicken Sie auf Domainnamenzugriffszuweisung erstellen.
4. Für Domainnamen-ARN wählen Sie den Domainnamen-ARN aus, den der API-Anbieter mit Ihnen geteilt hat.

Der Domänennamen-ARN erscheint möglicherweise nicht in der Dropdown-Liste. Sie können die AWS RAM Konsole verwenden, um Domainnamen anzuzeigen, die mit Ihnen geteilt wurden, und dann den Domainnamen ARN kopieren und in dieses Feld eingeben.

5. Wählen Sie für VPC-Endpunkt-ID die VPC-Endpunkt-ID aus, mit der Sie die Domainnamenzugriffszuweisung bilden möchten.
6. Klicken Sie auf Domainnamenzugriffszuweisung erstellen.

AWS CLI

Da private benutzerdefinierte Domainnamen nicht eindeutig sind, verknüpfen Sie Ihren VPC-Endpunkt mit dem eindeutigen benutzerdefinierten Domainnamen-ARN. Verwenden Sie einen der folgenden Befehle, um den Domänennamen-ARN zu finden.

1. AWS RAM

Der Befehl [list-resources](#) listet Ressourcen auf, die mit Ihnen gemeinsam genutzt werden. Um diesen Befehl verwenden AWS RAM zu können, muss der API-Anbieter zuvor seine private benutzerdefinierte Domain mit Ihnen geteilt haben.

```
aws ram list-resources \
  --resource-owner OTHER-ACCOUNTS \
  --region us-west-2
  --resource-type apigateway:Domainnames
```

API Gateway

Der folgende `get-domain-names` Befehl listet alle privaten benutzerdefinierten Domainnamen auf AWS-Konten, die anderen gehören und mit denen Sie Zugriffszuordnungen für Domainnamen bilden können.

```
aws apigateway get-domain-names \  
  --resource-owner OTHER_ACCOUNTS \  
  --region us-west-2
```

2. Nachdem Sie den ARN abgerufen haben, verwenden Sie API Gateway, um die Domainnamenzugriffszuweisung zwischen Ihrem VPC-Endpunkt und einem gemeinsam genutzten privaten benutzerdefinierten Domainnamen zu bilden. Verwenden Sie den folgenden `create-domain-name-access-association`-Befehl:

```
aws apigateway create-domain-name-access-association \  
  --access-association-source-type VPCE \  
  --access-association-source 'vpce-1a2b3c4d5e6f1a2b3' \  
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/  
private.example.com+abcd1234"
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "domainNameAccessAssociationARN": "arn:aws:apigateway:us-  
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com  
+abcd1234/vpcsource/vpce-abcd1234efg",  
  "accessAssociationSource": "vpce-1a2b3c4d5e6f1a2b3",  
  "accessAssociationSourceType": "VPCE",  
  "domainNameARN" : "arn:aws:apigateway:us-west-1:111122223333:/domainnames/  
private.example.com+a1b2c3"  
}
```

Nachdem Sie Ihren VPC-Endpunkt mit dem privaten benutzerdefinierten Domainnamen verknüpft haben, vergewissern Sie sich, dass Ihr API-Anbieter die Richtlinie für seinen privaten benutzerdefinierten Domainnamen aktualisiert hat, sodass Ihr VPC-Endpunkt seinen Domainnamen aufrufen kann. Weitere Informationen finden Sie unter [the section called “Anderen Konten Aufrufberechtigungen für Ihren privaten benutzerdefinierten Domainnamen gewähren”](#).

Erstellen einer von Route 53 gehosteten Zone

Sie müssen zunächst eine private gehostete Route 53-Zone erstellen, um den privaten benutzerdefinierten Domainnamen auflösen zu können. Eine gehostete Zone ist ein Container, der Informationen darüber enthält, wie Sie den Verkehr für eine Domain innerhalb einer oder mehrerer Domains weiterleiten möchten, VPCs ohne Ihre Ressourcen dem Internet auszusetzen. Weitere Informationen finden Sie unter [Arbeiten mit privat gehosteten Zonen](#).

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung [von finden Sie unter Creating a private hosted zone](#) im Amazon Route 53 Developer Guide.

Geben Sie unter Name den Namen des privaten benutzerdefinierten Domainnamens an. Verwenden Sie für VPC-ID die VPC, die den VPC-Endpunkt enthält, den Sie für Ihre Domainnamenzugriffszuweisung verwendet haben.

AWS CLI

Der folgende [create-hosted-zone](#)Befehl erstellt eine private gehostete Zone:

```
aws route53 create-hosted-zone --name private.example.com \
  --caller-reference 2014-04-01-18:47 \
  --hosted-zone-config Comment="command-line version",PrivateZone=true \
  --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

Die Ausgabe enthält die ID der gehosteten Zone. Die ID der gehosteten Zone brauchen Sie für die nachfolgenden Schritte.

Erstellen eines Route 53-DNS-Datensatzes

Nachdem Sie die gehostete Zone erstellt haben, erstellen Sie einen Datensatz, mit dem Sie die private benutzerdefinierte Domäne auflösen. In diesem Beispiel erstellen Sie einen A-Datensatztyp. Wenn Sie IPv6 für Ihren VPC-Endpunkt verwenden, erstellen Sie einen AAAA-Eintragstyp. Wenn Sie Dualstack für Ihren VPC-Endpunkt verwenden, erstellen Sie sowohl einen AAAA- als auch einen A-Datensatztyp.

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung von finden Sie unter [Weiterleiten von Datenverkehr an eine Amazon API Gateway Gateway-API mithilfe Ihres Domainnamens](#).

Verwenden Sie die Funktion Schnellerstellung und aktivieren Sie Alias. Für Endpunkt nehmen Sie den DNS-Namen des VPC-Endpunkts.

AWS CLI

Wenn Sie Ihre DNS-Datensätze so konfigurieren möchten, dass der private benutzerdefinierte Domainname dem Hostnamen der vorgegebenen gehosteten Zonen-ID zugewiesen wird, erstellen Sie zunächst eine JSON-Datei, die die Konfiguration für die Einrichtung eines DNS-Datensatzes für den privaten Domainnamen enthält.

Im Folgenden `setup-dns-record.json` wird gezeigt, wie Sie einen DNSA-Datensatz erstellen, um einen privaten benutzerdefinierten Domainnamen seinem privaten Hostnamen zuzuweisen. Sie geben den DNSName Ihrer VPC-DNS-ID und die Hosting-Zonen-ID an, die Sie oben erstellt haben.

```
{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z20JLYMU09EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Der folgende [change-resource-record-sets](#) Befehl erstellt einen DNS-Eintrag für den privaten benutzerdefinierten Domainnamen:

```
aws route53 change-resource-record-sets \
  --hosted-zone-id ZABCEFG1234 \
  --change-batch file:///path/to/your/setup-dns-record.json
```

Ersetzen Sie die `hosted-zone-id` durch die Route-53-Hosted-Zone-ID des DNS-Datensatzes, die in Ihrem Konto eingerichtet wurde. Der `change-batch`-Parameterwert verweist auf eine JSON-Datei.

Nächste Schritte für einen API-Verbraucher

Sie können die private API jetzt in Ihrem eigenen AWS-Konto aufrufen. Verwenden Sie in Ihrer VPC den folgenden curl-Befehl, um auf Ihren privaten benutzerdefinierten Domainnamen zuzugreifen.

```
curl https://private.example.com/v1
```

Weitere Informationen zu anderen Möglichkeiten Ihre private API aufzurufen, finden Sie unter [the section called “Aufrufen einer privaten API mithilfe eines benutzerdefinierten Domainnamens”](#).

API-Verbraucher: Löschen Ihrer Domainnamenzugriffszuweisung zu einem privaten benutzerdefinierten Domainnamen

Wenn Sie ein API-Verbraucher sind, können Sie die Zugriffszuweisungsressource jederzeit löschen. Der API-Anbieter kann die Domainnamenzugriffszuweisung nicht für Sie löschen.

Wir empfehlen, eine Domainnamenzugriffszuweisung immer dann zu löschen, wenn Sie sie nicht mehr verwenden.

AWS-Managementkonsole

So löschen Sie die Domainnamenzugriffszuweisung

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway>.
2. Klicken Sie im Hauptnavigationsbereich auf Domainnamenzugriffszuweisungen.
3. Wählen Sie Ihre Domainnamenzugriffszuweisung aus und klicken Sie dann auf Löschen.
4. Bestätigen Sie Ihr Auswahl und klicken Sie auf Löschen.

AWS CLI

Der Befehl `delete-access-association` löscht die Zugriffszuweisung:

```
aws apigateway delete-domain-name-access-association \
  --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcsource/vpce-abcd1234efg'
```

Erstellen Sie einen benutzerdefinierten Domainnamen für den privaten Gebrauch APIs CloudFormation

Die folgende CloudFormation Beispielvorlage erstellt eine private API und einen privaten benutzerdefinierten Domainnamen, ordnet die private API dem benutzerdefinierten Domainnamen zu und erstellt dann eine Zuordnung für den Domainnamen-Zugriff. Sie müssen Ihren eigenen VPC-Endpunkt, Domainnamen und Zertifikat-ARN angeben.

Die folgenden Überlegungen könnten sich auf Ihre Verwendung von auswirken CloudFormation , um einen privaten benutzerdefinierten Domainnamen zu erstellen:

- Sie können eine Zuordnung für den Zugriff auf Domainnamen nicht ablehnen, indem Sie CloudFormation. Verwenden Sie stattdessen die AWS CLI, um eine Domainnamenzugriffszuweisung abzulehnen.
- Verwenden Sie die `AWS::ApiGateway::DomainNameV2` CloudFormation Eigenschaft, um einen privaten benutzerdefinierten Domainnamen zu erstellen.
- Verwenden Sie die `AWS::ApiGateway::BasePathMappingV2` CloudFormation Eigenschaft, um eine Basispfadzuordnung zu erstellen.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EndpointID:
    Type: String
    Default: vpce-abcd1234567efg
    Description: A VPC endpoint with enableDnsHostnames and enableDnsSupport set to true.
  DomainName:
    Type: String
    Default: private.example.com
    Description: A domain name that you own.
  CertificateArn:
    Type: String
    Default: arn:aws:acm:us-west-2:123456789:certificate/abcd-000-1234-0000-000000abcd
    Description: An ACM certificate that covers the domain name.
Resources:
  PrivateApi:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      EndpointConfiguration:
        Types:
```

```

    - PRIVATE
    VpcEndpointIds:
      - !Ref EndpointID
  Name: private-api
  Policy:
    Statement:
      - Action: 'execute-api:Invoke'
        Effect: Allow
        Principal: '*'
        Resource: 'execute-api:/*'
      - Action: 'execute-api:Invoke'
        Condition:
          StringNotEquals:
            'aws:SourceVpce': !Ref EndpointID
        Effect: Deny
        Principal: '*'
        Resource: 'execute-api:/*'
  Version: 2012-10-17
PrivateApiDeployment:
  Type: 'AWS::ApiGateway::Deployment'
  Properties:
    RestApiId: !Ref PrivateApi
    Description: Private API deployment
  DependsOn:
    - PrivateApiMethod
PrivateApiStage:
  Type: 'AWS::ApiGateway::Stage'
  Properties:
    RestApiId: !Ref PrivateApi
    DeploymentId: !Ref PrivateApiDeployment
    StageName: prod
PrivateApiMethod:
  Type: 'AWS::ApiGateway::Method'
  Properties:
    HttpMethod: ANY
    ResourceId: !GetAtt PrivateApi.RootResourceId
    RestApiId: !Ref PrivateApi
    AuthorizationType: NONE
    Integration:
      Type: MOCK
      RequestTemplates:
        application/json: "{\"statusCode\": 200}"
      IntegrationResponses:
        - StatusCode: '200'

```

```
MethodResponses:
  - StatusCode: '200'
PrivateDomainName:
  Type: AWS::ApiGateway::DomainNameV2
  Properties:
    DomainName: !Ref DomainName
    CertificateArn: !Ref CertificateArn
    EndpointConfiguration:
      Types:
        - PRIVATE
    SecurityPolicy: TLS_1_2
    Policy:
      Statement:
        - Action: 'execute-api:Invoke'
          Effect: Allow
          Principal: '*'
          Resource: 'execute-api:/*'
        - Action: 'execute-api:Invoke'
          Condition:
            StringNotEquals:
              'aws:SourceVpce': !Ref EndpointID
          Effect: Deny
          Principal: '*'
          Resource: 'execute-api:/*'
      Version: 2012-10-17
PrivateBasePathMapping:
  Type: AWS::ApiGateway::BasePathMappingV2
  DependsOn:
    - PrivateApiStage
  Properties:
    BasePath: prod
    DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
    RestApiId: !Ref PrivateApi
    Stage: prod
DomainNameAccessAssociation:
  Type: AWS::ApiGateway::DomainNameAccessAssociation
  Properties:
    DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
    AccessAssociationSource: !Ref EndpointID
    AccessAssociationSourceType: VPCE
```

Aufrufen einer privaten API

In einer VPC können Sie eine private API nur mithilfe eines VPC-Endpunkts aufrufen. Ihre private API muss über eine Ressourcenrichtlinie verfügen, die es bestimmten VPCs und VPC-Endpunkten ermöglicht, Ihre API aufzurufen.

Wenn Sie eine private API aufrufen, ohne einen benutzerdefinierten Domainnamen oder private DNS-Namen zu verwenden, und Ihr APIs oder Ihr Domainname eine Sicherheitsrichtlinie verwendet, die mit `beginntSecurityPolicy_`, müssen Sie den Endpunktzugriffsmodus auf `BASIC` einstellen. Weitere Informationen finden Sie unter [the section called “Endpunktzugriffsmodus”](#).

Aufrufen einer privaten API mithilfe eines benutzerdefinierten Domainnamens

Ihr VPC-Endpunkt benötigt eine Domainnamenzugriffszuweisung mit einem benutzerdefinierten Domainnamen, und der benutzerdefinierte Domainname muss den Zugriff für den VPC-Endpunkt erlauben, um eine private API aufzurufen. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen für private APIs”](#).

Es gibt keine Unterschiede zwischen dem Aufrufen eines privaten benutzerdefinierten Domainnamens in einer VPC in Ihrer eigenen AWS-Konto oder in einer anderen. AWS-Konto

Verwenden Ihres benutzerdefinierten Domainnamens

In Ihrer VPC können Sie Ihre API mit dem benutzerdefinierten Domainnamen aufrufen. Im folgenden Beispiel wird ein `curl`-Befehl zum Aufrufen Ihres privaten benutzerdefinierten Domainnamens verwendet:

```
curl https://private.example.com
```

Verwenden von endpunktspezifischen privaten DNS-Hostnamen

Sie können Ihre API aufrufen, indem Sie den benutzerdefinierten Domainnamen als Endpunktspezifischen privaten DNS-Hostnamen angeben.

```
curl https://private-dns-hostname.execute-api.region.vpce.amazonaws.com/basepath -H  
'Host:custom-domain-name'
```

Im folgenden Beispiel wird ein `curl`-Befehl zum Aufrufen Ihres benutzerdefinierten Domainnamens mithilfe des Endpunkt-spezifischen privaten DNS-Hostnamens verwendet:

```
curl https://vpce-123456-abc000.execute-api.us-east-2.vpce.amazonaws.com/test -H
'Host:private.example.com'
```

Aufrufen einer privaten API ohne Verwendung eines benutzerdefinierten Domainnamens

Sie müssen die DNS-Namen für Ihre API angeben, wenn Sie Ihre private API ohne eine Verwendung eines benutzerdefinierten Domainnamens aufrufen möchten. Im folgenden Verfahren sehen Sie, wie Sie Ihre DNS-Namen finden.

AWS-Managementkonsole

So finden Sie die DNS-Namen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter <https://console.aws.amazon.com/vpc/>.
2. Klicken Sie im Hauptnavigationsbereich auf Endpunkte und wählen Sie dann Ihren Schnittstellen-VPC-Endpunkt für API Gateway aus.
3. Im Bereich Details werden fünf Werte im Feld DNS-Namen angezeigt. Die ersten drei sind die öffentlichen DNS-Namen für Ihre API. Die anderen zwei sind die privaten DNS-Namen.

AWS CLI

Verwenden Sie den folgenden [describe-vpc-endpoints](#)Befehl, um Ihre DNS-Werte aufzulisten.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

Die ersten drei sind die öffentlichen DNS-Namen für Ihre API. Die anderen zwei sind die privaten DNS-Namen.

Zugriff auf eine private API mithilfe eines Route53-Alias

Sie können einen VPC-Endpunkt zu Ihrer privaten API zuordnen oder die Zuweisung aufheben. Weitere Informationen finden Sie unter [the section called “\(Optional\) Zuordnen oder Aufheben der Zuweisung eines VPC-Endpunkts zu einer privaten API”](#).

Nachdem Sie Ihre VPC-Endpunkte mit Ihrer privaten API verknüpft haben, können Sie die API mithilfe der folgenden Basis-URL aufrufen:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die GET /pets-Methode für die test-Stufe eingerichtet haben und Ihre REST-API-ID und VPC-Endpoint-ID ist vpce-01234567abcdef012, und Ihre Region ist us-west-2, dann können Sie Ihre API wie folgt aufrufen:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/test/pets
```

Aufrufen einer privaten API mithilfe von privaten DNS-Namen

Wenn Ihr privates DNS aktiviert ist, können Sie unter Verwendung des folgenden privaten DNS-Namens auf Ihre private API zugreifen:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

Die Basis-URL zum Aufrufen der API besitzt das folgende Format:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die GET /pets-Methode für die test-Stufe eingerichtet haben und Ihre REST-API-ID ist 01234567ab und Ihre Region ist us-west-2, können Sie Ihre private API aufrufen, indem Sie folgende URL in einen Browser eingeben:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Alternativ könnten Sie Ihre private API mit folgendem cURL-Befehl aufrufen:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Warning

Wenn Sie privates DNS für Ihren VPC-Endpoint aktivieren, können Sie nicht auf den Standard-Endpoint für den öffentlichen APIs Zugriff zugreifen. Weitere Informationen finden Sie unter [Warum kann ich von einem API Gateway-VPC-Endpoint keine Verbindung zu meiner öffentlichen API herstellen?](#).

Rufen Sie eine private API auf mit Direct Connect

Sie können Direct Connect damit eine dedizierte private Verbindung von einem lokalen Netzwerk zu Amazon VPC herstellen und über diese Verbindung mithilfe öffentlicher DNS-Namen auf Ihren privaten API-Endpunkt zugreifen.

Sie können auch private DNS-Namen verwenden, um von einem lokalen Netzwerk aus auf Ihre private API zuzugreifen, indem Sie einen Amazon Route 53 Resolver eingehenden Endpunkt einrichten und an diesen alle DNS-Abfragen des privaten DNS aus Ihrem Remote-Netzwerk weiterleiten. Weitere Informationen finden Sie unter [Weiterleiten eingehender DNS-Abfragen an Ihre VPCs](#) im Amazon Route 53-Entwicklerhandbuch.

Aufrufen einer privaten API mithilfe von Endpunkt-spezifischen öffentlichen DNS-Hostnamen

Sie können auf Ihre private API mit endpunktspezifischen DNS-Hostnamen zugreifen. Dabei handelt es sich um öffentliche DNS-Hostnamen mit der VPC-Endpunkt-ID oder der API-ID für Ihre private API.

Die generierte Basis-URL liegt im folgenden Format vor:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die GET /pets Methode für die test Phase eingerichtet haben und Ihre REST-API-ID abc1234, ihr öffentlicher DNS-Hostname und Ihre Region lautet evpce-def-01234567, könnten Sie Ihre private API mit ihrer VPCe ID aufrufen us-west-2, indem Sie den Host Header in einem cURL-Befehl verwenden:

```
curl -v https://vpce-def-01234567.execute-api.us-west-2.vpce.amazonaws.com/test/pets -H 'Host: abc1234.execute-api.us-west-2.amazonaws.com'
```

Alternativ können Sie über ihre API-ID mit dem x-apigw-api-id-Header in einem cURL-Befehl im folgenden Format auf Ihre private API zugreifen:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -H 'x-apigw-api-id:{api-id}'
```

REST APIs in API Gateway überwachen

In diesem Abschnitt erfahren Sie, wie Sie Ihre API mithilfe von CloudWatch-Metriken, CloudWatch Logs, Firehose und AWS X-Ray überwachen. Durch die Kombination von CloudWatch-

Ausführungsprotokollen und CloudWatch-Metriken können Sie Fehler und Ablaufverfolgungen der Ausführung protokollieren und die Leistung Ihrer API überwachen. Möglicherweise möchten Sie auch API-Aufrufe an Firehose protokollieren. Sie können auch AWS X-Ray verwenden, um Aufrufe über die nachgeschalteten Services zu verfolgen, aus denen Ihre API besteht.

 Note

API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:

- 413 Request Entity Too Large-Fehler
- 431 Request-Header-Fields-Too-Large-Fehler
- Übermäßige 429 Too Many Requests-Fehler
- Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
- Fehler der Serie 500, die durch interne Ausfälle verursacht werden

API Gateway generiert beim Testen einer REST-API-Methode keine Protokolle und Metriken. Die CloudWatch-Einträge werden simuliert. Weitere Informationen finden Sie unter [the section called “Testen einer REST-API-Methode mithilfe der Konsole”](#).

Themen

- [Überwachen Sie die REST-API-Ausführung mit CloudWatch Amazon-Metriken](#)
- [Richten Sie die CloudWatch Protokollierung für REST APIs in API Gateway ein](#)
- [Protokollierung von REST-API-Aufrufen unter Amazon Data Firehose im API Gateway](#)
- [Variablen für die Zugriffsprotokollierung in API Gateway](#)
- [Benutzeranforderungen für Ablaufverfolgung an REST-APIs mithilfe von X-Ray in API Gateway](#)

Überwachen Sie die REST-API-Ausführung mit CloudWatch Amazon-Metriken

Sie können die API-Ausführung überwachen CloudWatch, indem Sie die Rohdaten von API Gateway sammeln und in lesbare near-real-time Metriken umwandeln. Diese Statistiken werden für einen

Zeitraum von 15 Monaten aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt wird. Standardmäßig werden API-Gateway-Metrikdaten automatisch innerhalb von CloudWatch einer Minute gesendet. Weitere Informationen finden Sie unter [Was ist Amazon CloudWatch?](#) im CloudWatch Amazon-Benutzerhandbuch.

Die von API Gateway gemeldeten Metriken bieten Informationen, die Sie auf unterschiedliche Weise analysieren können. Die folgende Liste zeigt einige häufige Verwendungszwecke für die Metriken, die Vorschläge für den Einstieg sind:

- Überwachen Sie die IntegrationLatency-Metriken, um die Reaktionsfähigkeit des Backends zu messen.
- Überwachen Sie die Latency-Metriken, um die allgemeine Reaktionsfähigkeit Ihrer API-Aufrufe zu messen.
- Überwachen Sie die CacheHitCount- und CacheMissCount-Metriken, um die Cache-Kapazitäten zu optimieren und die gewünschte Leistung zu erreichen.

Themen

- [Amazon API Gateway-Dimensionen und -Metriken](#)
- [Anzeigen von CloudWatch-Metriken mit dem API-Dashboard in API Gateway](#)
- [API Gateway Gateway-Metriken in der CloudWatch Konsole anzeigen](#)
- [API Gateway Gateway-Protokollereignisse in der CloudWatch Konsole anzeigen](#)
- [Überwachungstools AWS für API Gateway](#)

Amazon API Gateway-Dimensionen und -Metriken

Die Metriken und Dimensionen, die API Gateway an Amazon sendet, CloudWatch sind unten aufgeführt. Weitere Informationen finden Sie unter [Überwachen Sie die REST-API-Ausführung mit CloudWatch Amazon-Metriken](#).

API Gateway-Metriken

Amazon API Gateway sendet Metrikdaten an CloudWatch jede Minute.

Der AWS/ApiGateway-Namespace enthält die folgenden Metriken.

Metrik	Beschreibung
4XXError	Die Anzahl der erfassten clientseitigen Fehler innerhalb eines bestimmten Zeitraums. API Gateway zählt modifizierte Gateway-Antwortstatuscodes als 4 XXError Fehler. Die Statistik Sum repräsentiert diese Metrik, also die Gesamtzahl der 4XXError-Fehler in dem angegebenen Zeitraum. Die Statistik Average repräsentiert die 4XXError-Fehlerrate, also die Gesamtzahl der 4XXError-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
5XXError	Die Anzahl der erfassten serverseitigen Fehler innerhalb eines bestimmten Zeitraums Die Statistik Sum repräsentiert diese Metrik, also die Gesamtzahl der 5XXError-Fehler in dem angegebenen Zeitraum. Die Statistik Average repräsentiert die 5XXError-Fehlerrate, also die Gesamtzahl der 5XXError-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
CacheHitCount	Die Anzahl aller vom API-Cache innerhalb eines bestimmten Zeitraums bedienten Anforderungen Die Statistik Sum repräsentiert diese Metrik, also die Gesamtzahl der Cache-Zugriffe innerhalb des angegebenen Zeitraums. Die Statistik Average stellt die Cache-Zugriffsrate, also die Gesamtzahl der Cache-Zugriffe geteilt durch die Gesamtzahl der

Metrik	Beschreibung
	Anforderungen innerhalb des Zeitraums, dar. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
CacheMissCount	Die Anzahl aller vom Backend innerhalb eines bestimmten Zeitraums bedienten Anforderungen, wenn das API-Caching aktiviert ist Die Statistik Sum repräsentiert diese Metrik, also die Gesamtzahl der Cache-Fehler innerhalb des angegebenen Zeitraums. Die Statistik Average repräsentiert die Cache-Fehlerrate, also die Gesamtzahl der Cache-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
Count	Die Gesamtzahl der API-Anforderungen innerhalb eines bestimmten Zeitraums Die Statistik SampleCount stellt diese Metrik dar. Unit: Count
IntegrationLatency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage an das Backend weiterleitet und zu dem Zeitpunkt, zu dem es eine Antwort vom Backend erhält. Unit: Millisecond

Metrik	Beschreibung
Latency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage von einem Client empfängt, und dem Zeitpunkt, an dem es eine Antwort an den Client zurückgibt. Die Latenzzeit schließt die Integrationslatenzzeit und anderen API Gateway-Overhead ein. Unit: Millisecond

Dimensionen für Metriken

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

Note

API Gateway entfernt Nicht-ASCII-Zeichen aus der ApiName Dimension, bevor Metriken an CloudWatch gesendet werden. Wenn der APIName keine ASCII-Zeichen enthält, wird die API ID als ApiName verwendet.

Dimension	Beschreibung
ApiName	Filtert API Gateway-Metriken für die REST-API mit dem angegebenen API-Namen.
ApiName, Method, Resource, Stage	Filtert API Gateway-Metriken für die API-Methode mit dem angegebenen API-Namen, der Stufe, Ressource und Methode. API Gateway sendet diese Metriken nur, wenn Sie detaillierte CloudWatch Metriken explizit aktiviert haben. Wählen Sie in der Konsole eine Stufe aus, und klicken Sie dann unter Protokolle und Nachverfolgung auf Bearbeiten. Wählen Sie Detaillierte

Dimension	Beschreibung
	Metriken aus und klicken Sie dann auf Änderungen speichern. Alternativ können Sie den AWS CLI Befehl <code>aws apigateway update-stage</code> aufrufen, um die <code>metricsEnabled</code> Eigenschaft zu aktualisieren. <code>true</code> Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen Gebühren belastet. Preisinformationen finden Sie unter CloudWatchAmazon-Preise.
ApiName, Stage	Filtert API Gateway-Metriken für die API-Stufe <code>ApiName</code> mit dem angegebenen API-Namen und der Stufe.

Anzeigen von CloudWatch-Metriken mit dem API-Dashboard in API Gateway

Sie können das API-Dashboard in der API Gateway-Konsole verwenden, um die CloudWatch-Metriken Ihrer bereitgestellten API in API Gateway anzuzeigen. Diese werden als eine Zusammenfassung der API-Aktivität im Zeitverlauf angezeigt.

Themen

- [Voraussetzungen](#)
- [Untersuchen von API-Aktivitäten im Dashboard](#)

Voraussetzungen

1. Sie müssen über eine in API Gateway erstellte API verfügen. Folgen Sie den Anweisungen in [Entwickeln einer REST-API in API Gateway](#).
2. Sie müssen die API mindestens einmal bereitgestellt haben. Folgen Sie den Anweisungen in [REST-APIs in API Gateway bereitstellen](#).

Untersuchen von API-Aktivitäten im Dashboard

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.

2. Wählen Sie eine API aus.
3. Wählen Sie im Haupt-Navigationsbereich Dashboard aus.
4. Wählen Sie für Stufe die gewünschte Stufe aus.
5. Wählen Sie Datumsbereich aus, um einen Datumsbereich anzugeben.
6. Aktualisieren Sie bei Bedarf einzelne Metriken in separaten Grafiken mit den Titeln API-Aufrufe, Latenz, Integrationslatenz, Latenz, 4xx-Fehler und 5xx-Fehler.

 Tip

Um die CloudWatch-Metriken auf Methodenebene zu untersuchen, stellen Sie sicher, dass Sie CloudWatch Logs auf einer Methodenebene aktiviert haben. Weitere Informationen zum Einrichten der Protokollierung auf Methodenebene finden Sie unter [Überschreiben der Einstellungen auf Stufenebene](#).

API Gateway Gateway-Metriken in der CloudWatch Konsole anzeigen

Metriken werden zunächst nach dem Service-Namespace und anschließend nach den verschiedenen Dimensionskombinationen in den einzelnen Namespaces gruppiert. Aktivieren Sie „Detaillierte Metriken“, um die Metriken auf Methodenebene für Ihre API anzuzeigen. Weitere Informationen finden Sie unter [the section called “Aktualisieren von Stufeneinstellungen”](#).

So zeigen Sie API Gateway Gateway-Metriken mit der CloudWatch Konsole an

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Ändern Sie, falls erforderlich, die AWS-Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden.
3. Wählen Sie im Navigationsbereich Metrics aus.
4. Klicken Sie auf der Registerkarte All metrics auf API Gateway.
5. Klicken Sie auf den Bereich By Stage, um Metriken nach Phase anzuzeigen. Wählen Sie dann Ihre Namen APIs und die Namen der Metriken aus.
6. Klicken Sie auf den Bereich By Api Name, um Metriken für eine bestimmte API anzuzeigen. Wählen Sie dann Ihre Namen APIs und die Namen der Metriken aus.

So zeigen Sie Metriken mit der AWS CLI an

1. Verwenden Sie den folgenden [list-metrics](#)-Befehl, um Metriken aufzuführen:

```
aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
```

Nachdem Sie eine Metrik erstellt haben, kann es bis zu 15 Minuten dauern, bis die Metrik angezeigt wird. Verwenden Sie [get-metric-data](#) oder, um Metrikstatistiken schneller zu sehen [get-metric-statistics](#).

2. Verwenden Sie den folgenden [get-metrics-statistics](#)-Befehl, um den Durchschnitt über einen bestimmten Zeitraum in Intervallen von 5 Minuten anzuzeigen:

```
aws cloudwatch get-metric-statistics --namespace AWS/ApiGateway --metric-name Count  
--start-time 2011-10-03T23:00:00Z --end-time 2017-10-05T23:00:00Z --period 300 --  
statistics Average
```

API Gateway Gateway-Protokollereignisse in der CloudWatch Konsole anzeigen

Im folgenden Abschnitt werden die erforderlichen Voraussetzungen und die Anzeige von API-Gateway-Protokollereignissen in der CloudWatch Konsole erläutert.

Voraussetzungen

1. Sie müssen über eine in API Gateway erstellte API verfügen. Folgen Sie den Anweisungen in [Entwickeln einer REST-API in API Gateway](#).
2. Sie müssen die API mindestens einmal in der API bereitgestellt haben. Befolgen Sie die Anweisungen unter [REST-APIs in API Gateway bereitstellen](#) und [Aufrufen von REST-APIs in API Gateway](#).
3. Sie müssen CloudWatch Logs für eine Phase aktiviert haben. Folgen Sie den Anweisungen in [Richten Sie die CloudWatch Protokollierung für REST APIs in API Gateway ein](#).

Um protokollierte API-Anfragen und -Antworten mit der CloudWatch Konsole anzuzeigen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Ändern Sie, falls erforderlich, die AWS-Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden. Weitere Informationen finden Sie unter [Regionen und Endpunkte](#).

3. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
4. Wählen Sie in der Tabelle Protokollgruppen eine Protokollgruppe mit dem Namen API-Gateway-Execution-Logs_ `{{stage-name}}` aus. `rest-api-id`
5. Wählen Sie in der Tabelle Log Streams einen Protokoll-Stream aus. Sie können den gewünschten Protokoll-Stream anhand des Zeitstempels finden.
6. Klicken Sie auf Text, um den reinen Text anzuzeigen, oder klicken Sie auf Row, um das Ereignis zeilenweise anzuzeigen.

Important

CloudWatch ermöglicht das Löschen von Protokollgruppen oder Streams. Löschen Sie API Gateway-API-Protokollgruppen oder -Streams nicht manuell. Lassen Sie API Gateway diese Ressourcen verwalten. Wenn Sie Protokollgruppen oder Streams manuell löschen, werden API-Anforderungen und -Antworten möglicherweise nicht protokolliert. In diesem Fall können Sie die gesamte Protokollgruppe für die API löschen und die API erneut bereitstellen. Dies liegt daran, dass API Gateway Protokollgruppen oder -Streams einer API-Stufe zum Zeitpunkt der Bereitstellung erstellt.

Überwachungstools AWS für API Gateway

AWS bietet verschiedene Tools, mit denen Sie API Gateway überwachen können. Sie können einige dieser Tools für die automatische Überwachung konfigurieren, während andere manuellen Eingriff erfordern. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Automatisierte Überwachungstools in AWS

Sie können die folgenden automatisierten Tools zur Überwachung von API Gateway verwenden und auftretende Probleme melden:

- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (Amazon SNS) - Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Status befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein.

Weitere Informationen finden Sie unter [Überwachen Sie die REST-API-Ausführung mit CloudWatch Amazon-Metriken](#).

- Amazon CloudWatch Logs — Überwachen, speichern und greifen Sie auf Ihre Protokolldateien aus AWS CloudTrail oder anderen Quellen zu. Weitere Informationen finden Sie unter [Was ist CloudWatch Logs?](#) im CloudWatch Amazon-Benutzerhandbuch.
- Amazon EventBridge (früher CloudWatch Events genannt) — Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder Streams weiter, um Änderungen vorzunehmen, Statusinformationen zu erfassen und Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden Sie unter [Was ist Amazon EventBridge?](#) im EventBridge Benutzerhandbuch.
- AWS CloudTrail Protokollüberwachung — Teilen Sie Protokolldateien zwischen Konten, überwachen CloudTrail Sie Protokolldateien in Echtzeit, indem Sie sie an CloudWatch Logs senden, schreiben Sie Anwendungen zur Protokollverarbeitung in Java und stellen Sie sicher, dass sich Ihre Protokolldateien nach der Lieferung von nicht geändert haben CloudTrail. Weitere Informationen finden Sie unter [Arbeiten mit CloudTrail Protokolldateien](#) im AWS CloudTrail Benutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung von API Gateway besteht darin, die Elemente, die von den CloudWatch Alarmen nicht abgedeckt werden, manuell zu überwachen. Das API Gateway und andere AWS Konsolen-Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung. CloudWatch Wir empfehlen, auch die Protokolldateien für die API-Ausführung zu überprüfen.

- Das API Gateway-Dashboard zeigt die folgenden Statistiken für eine bestimmte API-Stufe in einem bestimmten Zeitraum:
 - API Calls
 - Cache Hit, nur wenn das API-Caching aktiviert ist.
 - Cache Miss, nur wenn das API-Caching aktiviert ist.
 - Latency
 - Integration Latency
 - 4XX Error
 - 5XX Error
- Auf der CloudWatch Startseite wird Folgendes angezeigt:
 - Aktuelle Alarmer und Status

- Diagramme mit Alarmen und Ressourcen
- Servicestatus

Darüber hinaus können CloudWatch Sie Folgendes verwenden:

- Erstellen [angepasster Dashboards](#) zur Überwachung der gewünschten Services.
- Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
- Suchen und durchsuchen Sie alle Ihre AWS Ressourcenmetriken
- Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

CloudWatch Alarme zur Überwachung des API Gateway erstellen

Sie können einen CloudWatch Alarm erstellen, der eine Amazon SNS SNS-Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum und führt eine oder mehrere Aktionen durch, die vom Wert der Metrik im Vergleich zu einem festgelegten Schwellenwert in einer Reihe von Zeiträumen abhängt. Die Aktion ist eine Benachrichtigung, die an ein Amazon SNS-Thema oder eine Auto Scaling-Richtlinie gesendet wird. Bei Alarmen werden nur Aktionen für anhaltende Statusänderungen ausgelöst. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Zustand muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein.

Richten Sie die CloudWatch Protokollierung für REST APIs in API Gateway ein

Um Probleme im Zusammenhang mit der Ausführung von Anfragen oder dem Client-Zugriff auf Ihre API zu beheben, können Sie Amazon CloudWatch Logs zur Protokollierung von API-Aufrufen aktivieren. Weitere Informationen zu finden Sie CloudWatch unter [the section called “CloudWatch Metriken”](#).

CloudWatch Protokollformate für API Gateway

Es gibt zwei Arten der API-Anmeldung CloudWatch: Ausführungsprotokollierung und Zugriffsprotokollierung. Bei der Ausführungsprotokollierung verwaltet API Gateway die CloudWatch Protokolle. Der Prozess umfasst das Erstellen von Protokollgruppen und -Streams sowie die Meldung der Anforderungen und Antworten jedes Aufrufers an Protokoll-Streams.

Die Protokolldaten umfassen Fehler oder Ablaufverfolgungen der Ausführung (z. B. Parameterwerte oder Nutzlasten von Anforderungen oder Antworten), von Lambda-Genehmigern (ehemals als benutzerdefinierte Genehmiger bezeichnet) verwendete Daten, Angaben darüber, ob API-Schlüssel erforderlich sind, Informationen dazu, ob Nutzungspläne aktiviert sind, und mehr. API Gateway redigiert Genehmigungs-Header, API-Schlüsselwerte und ähnliche sensible Anforderungsparameter aus den protokollierten Daten.

Zur Verbesserung Ihrer Sicherheitslage empfehlen wir Ihnen, die Ausführungsprotokollierung auf der ERROR- oder INFO-Ebene zu verwenden. Dies kann erforderlich sein, um verschiedene Compliance-Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

Wenn Sie eine API bereitstellen, erstellt API Gateway eine Protokollgruppe und -Streams unter der Protokollgruppe. Die Protokollgruppe wird nach dem API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}-Format benannt. In jeder Protokollgruppe, werden die Protokolle weiter in Protokoll-Streams unterteilt, die bei Meldung von Protokolldaten nach Last Event Time sortiert werden.

In der Zugriffsprotokollierung protokollieren Sie, als API-Entwickler, wer auf Ihre API zugegriffen hat und wie der Aufrufer auf die API zugegriffen hat. Erstellen Sie eine eigene Protokollgruppe oder wählen Sie eine vorhandene Protokollgruppe aus, die von API Gateway verwaltet werden kann. Wählen Sie [\\$context](#)-Variablen, ein Protokollformat und eine Protokollgruppe als Ziel aus, um die Zugriffsdetails anzugeben.

Das Zugriffsprotokoll-Format muss mindestens `$context.requestId` oder `$context.extendedRequestId` enthalten. Es hat sich bewährt, `$context.requestId` und `$context.extendedRequestId` in das Protokollformat aufzunehmen.

`$context.requestId`

Hiermit wird der Wert in der `x-amzn-RequestId`-Kopfzeile protokolliert. Clients können den Wert im `x-amzn-RequestId`-Header durch einen Wert im Format einer UUID (Universally Unique Identifier) überschreiben. API Gateway gibt diese Anforderungs-ID im `x-amzn-RequestId`-Antwort-Header zurück. API Gateway ersetzt überschriebene Anfragen IDs , die nicht das Format einer UUID haben, durch **UUID_REPLACED_INVALID_REQUEST_ID** in Ihren Zugriffsprotokollen.

`$context.extendedRequestId`

Die `extendedRequestId` ist eine eindeutige, von API Gateway generierte Kennung. API Gateway gibt diese Anforderungs-ID im `x-amz-apigw-id`-Antwort-Header zurück. Ein API-Aufrufer

kann diese Anforderungs-ID nicht bereitstellen oder überschreiben. Möglicherweise müssen Sie diesen Wert dem AWS Support zur Verfügung stellen, um die Fehlerbehebung für Ihre API zu unterstützen. Weitere Informationen finden Sie unter [the section called “Variablen für die Zugriffsprotokollierung in API Gateway”](#).

Wählen Sie ein Protokollformat aus, das auch von Ihrem analytischen Backend genutzt wird, z. B. [Common Log Format](#) (CLF), JSON, XML, oder CSV. Anschließend können Sie die Zugriffsprotokolle direkt eingeben und Ihre Metriken berechnen und rendern lassen. [Um das Protokollformat zu definieren, legen Sie den ARN der Protokollgruppe in der Eigenschaft accessLogSettings/DestinationArn auf der Bühne fest](#). Sie können einen Protokollgruppen-ARN in der CloudWatch Konsole abrufen. Um das Format des Zugriffsprotokolls zu definieren, legen Sie in der Eigenschaft [accessLogSetting/format auf der Bühne ein ausgewähltes Format](#) fest.

Beispiele für einige häufig verwendete Zugriffsprotokollformate werden in der API Gateway-Konsole dargestellt und wie folgt aufgeführt.

- CLF ([Common Log Format](#)):

```
$context.identity.sourceIp $context.identity.caller $context.identity.user
[$context.requestTime]"$context.httpMethod $context.resourcePath
$context.protocol" $context.status $context.responseLength $context.requestId
$context.extendedRequestId
```

- JSON:

```
{ "requestId":"$context.requestId",
  "extendedRequestId":"$context.extendedRequestId","ip": "$context.identity.sourceIp",
  "caller":"$context.identity.caller", "user":"$context.identity.user",
  "requestTime":"$context.requestTime", "httpMethod":"$context.httpMethod",
  "resourcePath":"$context.resourcePath", "status":"$context.status",
  "protocol":"$context.protocol", "responseLength":"$context.responseLength" }
```

- XML:

```
<request id="$context.requestId"> <extendedRequestId>$context.extendedRequestId</
extendedRequestId> <ip>$context.identity.sourceIp</ip> <caller>
$context.identity.caller</caller> <user>$context.identity.user</user> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
<resourcePath>$context.resourcePath</resourcePath> <status>$context.status</status>
```

```
<protocol>$context.protocol</protocol> <responseLength>$context.responseLength</responseLength> </request>
```

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,  
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,  
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Berechtigungen für die Protokollierung CloudWatch

Um CloudWatch Logs zu aktivieren, müssen Sie API Gateway die Erlaubnis erteilen, Logs CloudWatch für Ihr Konto zu lesen und zu schreiben. [Amazon APIGateway PushToCloudWatchLogs](#) verfügt über alle erforderlichen Berechtigungen.

Note

API Gateway ruft AWS -Security-Token-Service auf, um die IAM-Rolle zu übernehmen. Stellen Sie also sicher, dass diese für die Region aktiviert AWS STS ist. Weitere Informationen finden Sie unter [AWS STS in einer AWS Region verwalten](#).

[Um Ihrem Konto diese Berechtigungen zu gewähren, erstellen Sie eine IAM-Rolle `apigateway.amazonaws.com` als vertrauenswürdige Entität, fügen Sie der IAM-Rolle die obige Richtlinie hinzu und legen Sie den ARN der IAM-Rolle in der `cloudWatchRole` Eigenschaft `Arn` in Ihrem Konto fest.](#) Sie müssen die Eigenschaft [cloudWatchRoleArn](#) für jede AWS Region, in der Sie Logs aktivieren möchten, separat festlegen. CloudWatch

Wenn Sie beim Einstellen des ARN für die IAM-Rolle eine Fehlermeldung erhalten, überprüfen Sie Ihre AWS -Security-Token-Service Kontoeinstellungen, um sicherzustellen, dass diese in der Region, die Sie verwenden, aktiviert AWS STS ist. Weitere Informationen zur Aktivierung AWS STS finden Sie im IAM-Benutzerhandbuch unter [AWS STS in einer AWS Region verwalten](#).

CloudWatch API-Protokollierung mit der API Gateway Gateway-Konsole einrichten

Um die CloudWatch API-Protokollierung einzurichten, müssen Sie die API in einer Phase bereitgestellt haben. Sie müssen auch [einen entsprechenden CloudWatch Logs-Rollen-ARN](#) für Ihr Konto konfiguriert haben.

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
 2. Wählen Sie im Hauptnavigationsbereich Settings (Einstellungen) und dann unter Logging (Protokollierung) die Option Edit (Bearbeiten) aus.
 3. Geben Sie für die CloudWatch Protokollrolle ARN einen ARN einer IAM-Rolle mit den entsprechenden Berechtigungen ein. Sie müssen dies einmal für jeden Vorgang tun AWS-Konto, der APIs mithilfe von API Gateway erstellt wurde.
 4. Wählen Sie im Hauptnavigationsbereich eine der folgenden Optionen aus APIs, und führen Sie dann eine der folgenden Aktionen aus:
 - a. Wählen Sie eine vorhandene API und anschließend eine Stufe aus.
 - b. Erstellen Sie eine API und stellen Sie diese dann einer Stufe bereit.
 5. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
 6. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
 7. So aktivieren Sie die Ausführungsprotokollierung:
 - a. Wählen Sie im Dropdownmenü „Protokolle“ eine CloudWatch Protokollierungsebene aus. Die folgenden Protokollstufen sind verfügbar:
 - Aus - Protokollierung ist für diese Stufe nicht aktiviert.
 - Nur Fehler - Protokollierung ist nur für Fehler aktiviert.
 - Fehler und Informationsprotokolle - Protokollierung ist für alle Ereignisse aktiviert.
 - b. (Optional) Wählen Sie Datenablaufverfolgung aus, um die Protokollierung der Datenablaufverfolgung für Ihre Stufe zu aktivieren. Dies kann bei der Fehlerbehebung nützlich sein APIs, kann jedoch dazu führen, dass sensible Daten protokolliert werden.
-  **Note**

Wir empfehlen, die Datenverfolgung nicht für die Produktion APIs zu verwenden.
- c. (Optional) Wählen Sie Detaillierte Metriken aus, um detaillierte CloudWatch Metriken zu aktivieren.

Weitere Informationen zu CloudWatch Metriken finden Sie unter [the section called “CloudWatch Metriken”](#).

8. So aktivieren Sie die Zugriffsprotokollierung:
 - a. Aktivieren Sie die Option Custom access logging (Benutzerdefinierte Zugriffsprotokollierung).
 - b. Geben Sie den ARN einer Protokollgruppe in Access Log Destination ARN (Ziel-ARN des Zugriffsprotokolls) ein. Das ARN-Format ist `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.
 - c. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Sie können zwischen CLF, JSON, XML oder CSV wählen. Weitere Informationen zu Beispielprotokollformaten finden Sie unter [the section called "CloudWatch Protokollformate für API Gateway"](#).
9. Wählen Sie Änderungen speichern aus.

 Note

Sie können die Ausführungs- und Zugriffsprotokollierung unabhängig voneinander aktivieren.

API Gateway kann nun Anforderungen an Ihre API protokollieren. Sie müssen die API nicht erneut bereitstellen, wenn Sie die Stufeneinstellungen, Protokolle oder Stufenvariablen aktualisieren.

Richten Sie die CloudWatch API-Protokollierung ein mit CloudFormation

Verwenden Sie die folgende CloudFormation Beispielvorlage, um eine Amazon CloudWatch Logs-Protokollgruppe zu erstellen und die Ausführungs- und Zugriffsprotokollierung für eine Phase zu konfigurieren. Um CloudWatch Logs zu aktivieren, müssen Sie API Gateway die Erlaubnis erteilen, Logs CloudWatch für Ihr Konto zu lesen und zu schreiben. Weitere Informationen finden Sie unter [Konto mit IAM-Rolle verknüpfen](#) im AWS CloudFormation -Benutzerhandbuch.

```
TestStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: test
    RestApiId: !Ref MyAPI
    DeploymentId: !Ref Deployment
    Description: "test stage description"
    MethodSettings:
      - ResourcePath: "/"
        HttpMethod: "*"
        LoggingLevel: INFO
```

```
AccessLogSetting:
  DestinationArn: !GetAtt MyLogGroup.Arn
  Format: $context.extendedRequestId $context.identity.sourceIp
    $context.identity.caller $context.identity.user [$context.requestTime]
    "$context.httpMethod $context.resourcePath $context.protocol" $context.status
    $context.responseLength $context.requestId
  MyLogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: !Join
        - '-'
        - - !Ref MyAPI
          - access-logs
```

Protokollierung von REST-API-Aufrufen unter Amazon Data Firehose im API Gateway

Sie können API-Aufrufe unter Amazon Data Firehose protokollieren, um Probleme im Zusammenhang mit dem Client-Zugriff auf Ihre API zu beheben. Weitere Informationen zu Firehose finden Sie unter [Was ist Amazon Data Firehose?](#).

Für die Zugriffsprotokollierung können Sie nur CloudWatch oder Firehose aktivieren — Sie können nicht beide aktivieren. Sie können jedoch die Ausführungsprotokollierung und Firehose für die Zugriffsprotokollierung aktivieren CloudWatch .

Themen

- [Firehose-Protokollformate für API Gateway](#)
- [Berechtigungen für die Firehose-Protokollierung](#)
- [Einrichten der Firehose-Zugriffsprotokollierung mithilfe der API-Gateway-Konsole](#)

Firehose-Protokollformate für API Gateway

Die Firehose-Protokollierung verwendet dasselbe Format wie die [CloudWatch Protokollierung](#).

Berechtigungen für die Firehose-Protokollierung

Wenn die Firehose-Zugriffsprotokollierung für eine Stufe aktiviert ist, erstellt API Gateway eine servicebezogene Rolle für Ihr Konto, sofern die Rolle nicht bereits vorhanden

ist. Die Rolle wird `AWSServiceRoleForAPIGateway` benannt und die verwaltete `APIGatewayServiceRolePolicy`-Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter [Verwenden serviceverknüpfter Rollen](#).

 Note

Der Name Ihres Firehose-Streams muss `amazon-apigateway-{your-stream-name}` lauten.

Einrichten der Firehose-Zugriffsprotokollierung mithilfe der API-Gateway-Konsole

Zum Einrichten der API-Protokollierung müssen Sie die API für eine Stufe bereitgestellt haben. Ein Firehose-Stream muss bereits erstellt worden sein.

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Führen Sie eine der folgenden Aktionen aus:
 - a. Wählen Sie eine vorhandene API und anschließend eine Stufe aus.
 - b. Erstellen Sie eine API und stellen Sie diese einer Stufe bereit.
3. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
4. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
5. So aktivieren Sie die Zugriffsprotokollierung für einen Firehose-Stream:
 - a. Aktivieren Sie die Option Custom access logging (Benutzerdefinierte Zugriffsprotokollierung).
 - b. Geben Sie den ARN eines Firehose-Streams unter ARN für Zugriffsprotokollziel ein. Das ARN-Format ist `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

 Note

Der Name Ihres Firehose-Streams muss `amazon-apigateway-{your-stream-name}` lauten.

- c. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Sie können zwischen CLF, JSON, XML oder CSV wählen. Weitere Informationen zu Beispielprotokollformaten finden Sie unter [the section called “CloudWatch Protokollformate für API Gateway”](#).
6. Wählen Sie Änderungen speichern aus.

API Gateway kann ab sofort API-Aufrufe unter Firehose protokollieren. Sie müssen die API nicht erneut bereitstellen, wenn Sie die Stufeneinstellungen, Protokolle oder Stufenvariablen aktualisieren.

Variablen für die Zugriffsprotokollierung in API Gateway

In der Zugriffsprotokollierung protokollieren Sie, als API-Entwickler, wer auf Ihre API zugegriffen hat und wie der Aufrufer auf die API zugegriffen hat. Erstellen Sie eine eigene Protokollgruppe oder wählen Sie eine vorhandene Protokollgruppe aus, die von API Gateway verwaltet werden kann. Um die Zugriffsdetails anzugeben, können Sie die folgenden `$context`-Variablen verwenden, bei denen die Groß- und Kleinschreibung beachtet werden muss.

Eine Liste der Referenzvariablen für Datentransformationen finden Sie unter [the section called “Variablen für Datentransformationen”](#).

Parameter	Description
<code>\$context.accountId</code>	Die AWS Konto-ID des API-Besitzers.
<code>\$context.apiId</code>	Die ID, die API Gateway Ihrer API zuweist.
<code>\$context.authorize.error</code>	Die Autorisierungsfehlermeldung.
<code>\$context.authorize.latency</code>	Die Autorisierungslatenz in ms
<code>\$context.authorize.status</code>	Der Statuscode, der von einem Autorisierungsversuch zurückgegeben wurde.
<code>\$context.authorizer.claims. <i>property</i></code>	Eine Eigenschaft der Claims, die aus dem Amazon Cognito-Benutzerpool zurückgegeben werden, nachdem der Methodenaufrufer erfolgreich authentifiziert wurde. Weitere Informationen finden Sie unter the section

Parameter	Description
	called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”.  Note Bei einem Aufruf von <code>\$context.authorizer.claims</code> wird null (0) zurückgegeben.
<code>\$context.authorizer.error</code>	Die von einem Genehmiger zurückgegebene Fehlermeldung.
<code>\$context.authorizer.integrationLatency</code>	Die Integrationslatenz des Genehmigers in Millisekunden
<code>\$context.authorizer.integrationStatus</code>	Der von einem Lambda-Genehmiger zurückgegebene Statuscode.
<code>\$context.authorizer.latency</code>	Der Genehmiger-Latenz in ms.
<code>\$context.authorizer.principalId</code>	Die ID des Prinzipalbenutzers in Verbindung mit dem Token, das vom Client gesendet und von einem API Gateway-Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bekannt) zurückgegeben wurde. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden” .

Parameter	Description
<code>\$context.authorizer.<i>property</i></code>	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der context-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende context-Zuweisung zurück: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Ein Aufruf von <code>\$context.authorizer.key</code> gibt die Zeichenfolge "value" zurück, ein Aufruf von <code>\$context.authorizer.numKey</code> gibt die Zeichenfolge "1" zurück und ein Aufruf von <code>\$context.authorizer.boolKey</code> gibt die Zeichenfolge "true" zurück. Denn <i>property</i> das einzige unterstützte Sonderzeichen ist der (_) Unterstrich. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden”.
<code>\$context.authorizer.requestId</code>	Die Anforderungs-ID des AWS Endpunkts.
<code>\$context.authorizer.status</code>	Der von einem Genehmiger zurückgegebene Statuscode.
<code>\$context.authenticate.error</code>	Die von einem Authentifizierungsversuch zurückgegebene Fehlermeldung.

Parameter	Description
<code>\$context.authenticate.latency</code>	Die Authentifizierungslatenz in ms
<code>\$context.authenticate.status</code>	Der Statuscode, der von einem Authentifizierungsversuch zurückgegeben wurde.
<code>\$context.awsEndpointRequestId</code>	Die Anforderungs-ID des AWS Endpunkts.
<code>\$context.cipherSuite</code>	Die Chiffre im IANA-Format, die während des TLS-Handshakes zwischen dem Client und dem API Gateway ausgehandelt wird.
<code>\$context.customDomain.basePathMatched</code>	Der Pfad für ein API-Mapping, mit dem eine eingehende Anforderung übereinstimmt. Gilt, wenn ein Client einen benutzerdefinierten Domain-Namen für den Zugriff auf eine API verwendet. Wenn ein Client beispielsweise eine Anforderung an <code>https://api.example.com/v1/orders/1234</code> sendet und die Anforderung dem API-Mapping mit dem Pfad <code>v1/orders</code> übereinstimmt, lautet der Wert <code>v1/orders</code> . Weitere Informationen hierzu finden Sie unter the section called “API-Mappings” .
<code>\$context.customDomain.routingRuleIdMatched</code>	Die Routing-Regel, mit der eine eingehende Anfrage übereinstimmt. Gilt, wenn ein Client einen benutzerdefinierten Domain-Namen für den Zugriff auf eine API verwendet. Weitere Informationen hierzu finden Sie unter the section called “Routing-Regeln” .
<code>\$context.deploymentId</code>	Die ID der API-Bereitstellung

Parameter	Description
<code>\$context.domainName</code>	Der zum Aufrufen der API verwendete vollständige Domännennamen. Dieser Wert sollte mit dem für den eingehenden Host-Header übereinstimmen.
<code>\$context.domainPrefix</code>	Das erste Label der <code>\$context.domainName</code> .
<code>\$context.endpointType</code>	Der Endpunkttyp der API.
<code>\$context.error.message</code>	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält. Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponseBody -Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und in der Zugriffsprotokollierung verwendet werden. Weitere Informationen erhalten Sie unter the section called “Kennzahlen” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen” .
<code>\$context.error.messageString</code>	Der Wert von <code>\$context.error.message</code> in Anführungszeichen, d. h. " <code>\$context.error.message</code> ".
<code>\$context.error.responseType</code>	Ein Typ von. GatewayResponse Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponseBody -Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und für die Zugriffsprotokollierung verwendet werden. Weitere Informationen erhalten Sie unter the section called “Kennzahlen” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen” .

Parameter	Description
<code>\$context.error.validationErrorString</code>	Eine Zeichenfolge mit einer detaillierten Validierungs-Fehlermeldung.
<code>\$context.extendedRequestId</code>	Die erweiterte ID, die API Gateway generiert und der API-Anfrage zuweist. Die erweiterte Anforderungs-ID enthält zusätzliche nützliche Informationen für Debugging und Fehlerbehebung.
<code>\$context.httpMethod</code>	Die verwendete HTTP-Methode. Gültige Werte sind: DELETE, GET, HEAD, OPTIONS, PATCH, POST und PUT.
<code>\$context.identity.accountId</code>	Die mit der Anfrage verknüpfte AWS Konto-ID.
<code>\$context.identity.apiKey</code>	Bei API-Methoden, für die ein API-Schlüssel erforderlich ist, ist diese Variable der API-Schlüssel für die Methodenanforderung. Bei Methoden, für die kein API-Schlüssel erforderlich ist, ist diese Variable nichtig. Weitere Informationen finden Sie unter the section called “Nutzungspläne” .
<code>\$context.identity.apiKeyId</code>	Die API-Schlüssel-ID für die API-Anforderung, falls ein API-Schlüssel erforderlich ist.
<code>\$context.identity.caller</code>	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Ressourcen unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Description
<code>\$context.identity.cognitoAuthenticationProvider</code>	Eine durch Komma getrennte Liste aller Amazon-Cognito-Authentifizierungsanbieter, die vom anfordernden Aufrufer verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, <code>cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Weitere Informationen zu verfügbaren Amazon-Cognito-Authentifizierungsanbietern finden Sie unter Verbundidentitäten verwenden im Amazon-Cognito-Entwicklerhandbuch.
<code>\$context.identity.cognitoAuthenticationType</code>	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind <code>authenticated</code> für authentifizierte Identitäten und <code>unauthenticated</code> für nicht authentifizierte Identitäten.
<code>\$context.identity.cognitoIdentityId</code>	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Description
<code>\$context.identity.cognitoId</code> <code>entityPoolId</code>	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.principalOrgId</code>	Die AWS -Organisations-ID .
<code>\$context.identity.sourceIp</code>	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an den API-Gateway-Endpunkt gesendet wird.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Das PEM-encoded Client-Zertifikat, das der Client bei der gegenseitigen TLS-Authentifizierung vorgelegt hat. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domain-Namens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.subjectDN</code>	Der Distinguished Name des Zertifikatantrags, den ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.issuerDN</code>	Der Distinguished Name des Ausstellers des Zertifikats, das ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.serialNumber</code>	Die Seriennummer des Zertifikats. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.

Parameter	Description
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Das Datum, vor dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Das Datum, nach dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domain-Namens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.vpcId</code>	Die VPC-ID der VPC, deren Anforderung an den API-Gateway-Endpunkt gesendet wird.
<code>\$context.identity.vpceId</code>	Die VPC-Endpunkt-ID des VPC-Endpunkts, dessen Anforderung an den API-Gateway-Endpunkt gesendet wird. Diese ist nur vorhanden, wenn Ihre API privat ist.
<code>\$context.identity.user</code>	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Ressourcen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.userAgent</code>	Die User-Agent -Kopfzeile des API-Aufrufers.
<code>\$context.identity.userArn</code>	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.integration.error</code>	Die von einer Integration zurückgegebene Fehlermeldung.

Parameter	Description
<code>\$context.integration.integrationStatus</code>	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.
<code>\$context.integration.latency</code>	Die Integrationslatenz in Millisekunden. Äquivalent mit <code>\$context.integrationLatency</code> .
<code>\$context.integration.requestId</code>	Die AWS Anforderungs-ID des Endpunkts . Äquivalent mit <code>\$context.awsEndpointRequestId</code> .
<code>\$context.integration.responseTransferMode</code>	Der Antwortübertragungsmodus Ihrer Integration. Dies kann entweder BUFFERED oder STREAMED sein.
<code>\$context.integration.status</code>	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird.
<code>\$context.integration.timeToAllHeaders</code>	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway die Integrationsverbindung herstellt , und dem Empfang aller Integrationsantwort-Header vom Client.
<code>\$context.integration.timeToFirstContent</code>	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway die Integrationsverbindung herstellt, und dem Empfang der ersten Inhaltsbytes.
<code>\$context.integrationLatency</code>	Die Integrationslatenz in Millisekunden.

Parameter	Description
<code>\$context.integrationStatus</code>	Für die Lambda-Proxyintegration stellt dieser Parameter den Statuscode dar, der vom Lambda-Funktionscode zurückgegeben wurde AWS Lambda, nicht aus dem Back-End-Lambda-Funktionscode.
<code>\$context.isCanaryRequest</code>	Gibt <code>true</code> zurück, wenn die Anforderung an den Canary gerichtet war oder <code>false</code> , wenn die Anforderung nicht an den Canary ging. Dies ist nur vorhanden, wenn Sie einen Canary aktiviert haben.
<code>\$context.path</code>	Der Anforderungspfad. Bei einer Nicht-Proxy-Anforderungs-URL von <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> lautet der <code>\$context.path</code> -Wert beispielsweise <code>{stage}/root/child</code> .
<code>\$context.protocol</code>	Das Anforderungsprotokoll ist z. B. HTTP/1.1.

 Note

API Gateway-APIs können HTTP/2 Anfragen annehmen, aber API Gateway sendet Anfragen an Backend-Integrationen über HTTP/1.1. Daher wird das Anforderungsprotokoll so protokolliert, HTTP/1.1 als ob ein Client eine Anfrage sendet, die HTTP/2

Parameter	Description
<code>\$context.requestId</code>	Eine ID für die Anforderung. Clients können diese Anforderungs-ID überschreiben. Verwenden von <code>\$context.requestId</code> für eine eindeutige Anforderungs-ID, die API Gateway generiert.
<code>\$context.requestOverride.header. <i>header_name</i></code>	Der Anforderungs-Header-Override. Wenn dieser Parameter definiert ist, enthält er die Header, die statt der HTTP Header, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestOverride.path. <i>path_name</i></code>	Der Anforderungspfad-Override. Wenn dieser Parameter definiert ist, enthält er den Anforderungspfad, der statt der URL-Pfadparameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestOverride.querystring. <i>querystring_name</i></code>	Der Abfragestring-Override. Wenn dieser Parameter definiert ist, enthält er die Abfragestrings, die statt der URL-Abfragestring-Parameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .

Parameter	Description
<code>\$context.responseLatency</code>	Die Antwortlatenz in Millisekunden.
<code>\$context.responseLength</code>	Die Länge der Antwortnutzlast in Byte.
<code>\$context.responseOverride.header. <i>header_name</i></code>	Der Antwort-Header-Override. Wenn dieser Parameter definiert ist, enthält er den Header, der anstelle des Antwort-Headers, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.responseOverride.status</code>	Der Antwortstatuscode-Override. Wenn dieser Parameter definiert ist, enthält er den Statuscode, der anstelle des Methoden-Antwortstatus, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Überschreiben der Anforderungs- und Antwortparameter sowie Statuscodes Ihrer API für REST-APIs in API Gateway .
<code>\$context.requestTime</code>	Die Anforderungszeit im CLF -Format (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	Die Anforderungszeit im Epoch -Format in Millisekunden.
<code>\$context.resourceId</code>	Der Bezeichner, den API Gateway Ihrer Ressource zuweist.

Parameter	Description
<code>\$context.resourcePath</code>	Der Pfad zu Ihrer Ressource. Beim Nicht-Proxy-Anforderungs-URI von <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> lautet der <code>\$context.resourcePath</code> - Wert beispielsweise <code>/root/child</code> . Weitere Informationen finden Sie unter Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen .
<code>\$context.stage</code>	Die Bereitstellungsstufe der API-Anforderung (z. B. Beta oder Prod).
<code>\$context.status</code>	Der Status der Methodenantwort.
<code>\$context.tlsVersion</code>	Die TLS-Version, die während des TLS-Handshakes zwischen dem Client und dem API Gateway ausgehandelt wird.
<code>\$context.waf.error</code>	Die Fehlermeldung wurde von AWS WAF zurückgegeben.
<code>\$context.waf.latency</code>	Die AWS WAF Latenz in ms.
<code>\$context.waf.status</code>	Der Statuscode wurde von zurückgegeben AWS WAF.
<code>\$context.xrayTraceId</code>	Die Trace-ID für den X-Ray Trace. Weitere Informationen finden Sie unter the section called “Einrichten AWS X-Ray” .
<code>\$context.wafResponseCode</code>	Die von AWS WAF empfangene Antwort: <code>WAF_ALLOW</code> oder <code>WAF_BLOCK</code> . Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called “AWS WAF” .

Parameter	Description
<code>\$context.webaclArn</code>	Vollständiger ARN der Web-Zugriffskontrollliste (Web-ACL), anhand deren entschieden wird, ob die Anforderung zugelassen oder blockiert wird. Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called “AWS WAF” .

Benutzeranforderungen für Ablaufverfolgung an REST-APIs mithilfe von X-Ray in API Gateway

Sie können [AWS X-Ray](#) verwenden, um Benutzeranforderungen auf ihrem Weg durch Ihre Amazon-API-Gateway-REST-APIs zu den zugrunde liegenden Diensten zu verfolgen und zu analysieren. API Gateway unterstützt die X-Ray-Ablaufverfolgung für alle API Gateway-REST-API-Endpunkttypen: Regional, Edge-optimiert und privat. Sie können X-Ray mit Amazon API Gateway in allen AWS-Regionen verwenden, in denen X-Ray verfügbar ist.

Da X-Ray Ihnen eine End-to-End-Ansicht einer gesamten Anforderung bietet, können Sie Latenzen in Ihren APIs und deren Backend-Services analysieren. Sie können eine X-Ray-Servicezuordnung verwenden, um die Latenz einer gesamten Anforderung und der der nachgeschalteten Services anzuzeigen, die in X-Ray integriert sind. Sie können auch Samplingregeln konfigurieren, um X-Ray mitzuteilen, welche Anforderungen mit welchen Abstraten gemäß den von Ihnen angegebenen Kriterien aufgezeichnet werden sollen.

Wenn Sie eine API Gateway-API von einem Service aufrufen, der bereits verfolgt wird, wird API Gateway die Ablaufverfolgung auch dann ausführen, wenn die X-Ray-Ablaufverfolgung in der API nicht aktiviert ist.

Sie können X-Ray für eine API-Stufe aktivieren, indem Sie die API Gateway-Konsole oder die API Gateway-API oder -CLI verwenden.

Themen

- [AWS X-Ray Mit API Gateway REST einrichten APIs](#)
- [Verwenden Sie AWS X-Ray Service Maps und Trace Views mit API Gateway](#)
- [AWS X-Ray Sampling-Regeln für API Gateway konfigurieren APIs](#)

- [AWS X-Ray Ablaufverfolgungen für Amazon API Gateway APIs](#)

AWS X-Ray Mit API Gateway REST einrichten APIs

In diesem Abschnitt finden Sie detaillierte Informationen zur Einrichtung [AWS X-Ray](#) mit API Gateway REST APIs.

Themen

- [X-Ray-Ablaufverfolgungsmodi für API Gateway](#)
- [Berechtigungen für die X-Ray-Ablaufverfolgung](#)
- [Aktivieren der X-Ray-Ablaufverfolgung in der API Gateway-Konsole](#)
- [Aktivieren der AWS X-Ray Ablaufverfolgung mit der API Gateway Gateway-CLI](#)

X-Ray-Ablaufverfolgungsmodi für API Gateway

Der Pfad einer Anforderung über Ihre Anwendung wird mit einer Nachverfolgungs-ID verfolgt. Eine Ablaufverfolgung erfasst alle von einer einzelnen Anforderung generierten Segmente; in der Regel handelt es sich dabei um eine HTTP GET- oder POST-Anforderung.

Es gibt zwei Ablaufverfolgungsmodi für eine API Gateway-API:

- **Passiv:** Dies ist die Standardeinstellung, wenn Sie die X-Ray-Ablaufverfolgung in einer API-Stufe nicht aktiviert haben. Bei diesem Ansatz wird die API Gateway-API nur nachverfolgt, wenn X-Ray für einen Upstream-Service aktiviert wurde.
- **Aktiv:** Wenn eine API Gateway-API-Stufe diese Einstellung aufweist, stellt API Gateway die API-Aufrufanfragen automatisch auf Basis des von X-Ray bereitgestellten Sampling-Algorithmus zusammen.

Wenn die aktive Ablaufverfolgung in einer Stufe aktiviert ist, wird API Gateway in Ihrem Konto eine servicebezogene Rolle erstellen, sofern die Rolle nicht bereits vorhanden ist. Die Rolle wird benannt `AWSServiceRoleForAPIGateway` und die `APIGatewayServiceRolePolicy` verwaltete Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter [Verwenden serviceverknüpfter Rollen](#).

 Note

X-Ray wendet einen Sampling-Algorithmus an, um eine effizient Ablaufverfolgung zu gewährleisten und gleichzeitig ein repräsentatives Beispiel für die Anforderungen bereitzustellen, die von Ihrer API bedient werden. Der Standardeinstellung für den Sampling-Algorithmus lautet 1 Anforderung pro Sekunde, wobei 5 Prozent der Anforderungen, die über dieses Limit hinausgehen, auch erfasst werden.

Sie können den Ablaufverfolgungsmodus für Ihre API ändern, indem Sie die API Gateway Gateway-Verwaltungskonsolle, die API Gateway Gateway-CLI oder ein AWS SDK verwenden.

Berechtigungen für die X-Ray-Ablaufverfolgung

Wenn Sie die X-Ray-Ablaufverfolgung in einer Stufe aktivieren, wird API Gateway in Ihrem Konto eine servicebezogene Rolle erstellen, sofern die Rolle nicht bereits vorhanden ist. Die Rolle wird benannt `AWSServiceRoleForAPIGateway` und die `APIGatewayServiceRolePolicy` verwaltete Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter [Verwenden serviceverknüpfter Rollen](#).

Aktivieren der X-Ray-Ablaufverfolgung in der API Gateway-Konsole

Sie können die Amazon API Gateway-Konsole verwenden, um die aktive Ablaufverfolgung in einer API-Stufe zu aktivieren.

Diese Schritte gehen davon aus, dass Sie die API bereits in einer Stufe bereitgestellt haben.

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre API und dann im Hauptnavigationsbereich die Option Stages (Stufe).
3. Wählen Sie im Bereich Stages (Stufen) eine Stufe aus.
4. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
5. Sie können die aktive X-Ray-Nachverfolgung aktivieren, indem Sie X-Ray tracing (X-Ray-Nachverfolgung) auswählen, um die X-Ray-Nachverfolgung zu aktivieren.
6. Wählen Sie Änderungen speichern aus.

Nachdem Sie X-Ray für Ihre API-Stufe aktiviert haben, können Sie die Ablaufverfolgungen und Service-Übersichten mithilfe der X-Ray-Verwaltungskonsole anzeigen.

Aktivieren der AWS X-Ray Ablaufverfolgung mit der API Gateway Gateway-CLI

Der folgende [create-stage](#)-Befehl erstellt eine Stufe mit aktiver X-Ray-Ablaufverfolgung:

```
aws apigateway create-stage \  
  --rest-api-id rest-api-id \  
  --stage-name stage-name \  
  --deployment-id deployment-id \  
  --region region \  
  --tracing-enabled=true
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "tracingEnabled": true,  
  "stageName": stage-name,  
  "cacheClusterEnabled": false,  
  "cacheClusterStatus": "NOT_AVAILABLE",  
  "deploymentId": deployment-id,  
  "lastUpdatedDate": 1533849811,  
  "createdDate": 1533849811,  
  "methodSettings": {}  
}
```

Der folgende [create-stage](#)-Befehl erstellt eine Stufe ohne aktive X-Ray-Ablaufverfolgung:

```
aws apigateway create-stage \  
  --rest-api-id rest-api-id \  
  --stage-name stage-name \  
  --deployment-id deployment-id \  
  --region region \  
  --tracing-enabled=false
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "tracingEnabled": false,  
  "stageName": stage-name,
```

```
{
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": deployment-id,
  "lastUpdatedDate": 1533849811,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```

Der folgende [update-stage](#)-Befehl aktiviert die aktive X-Ray-Ablaufverfolgung für eine bereitgestellte API:

```
aws apigateway update-stage \
  --rest-api-id rest-api-id \
  --stage-name stage-name \
  --patch-operations op=replace,path=/tracingEnabled,value=true
```

Der folgende [update-stage](#)-Befehl deaktiviert die aktive X-Ray-Ablaufverfolgung für eine bereitgestellte API:

```
aws apigateway update-stage \
  --rest-api-id rest-api-id \
  --stage-name stage-name \
  --region region \
  --patch-operations op=replace,path=/tracingEnabled,value=false
```

Die Ausgabe sieht wie folgt aus:

```
{
  "tracingEnabled": false,
  "stageName": stage-name,
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": deployment-id,
  "lastUpdatedDate": 1533850033,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```

Sobald Sie X-Ray für Ihre API-Stufe aktiviert haben, verwenden Sie die X-Ray-CLI zum Abrufen von Ablaufverfolgungsinformationen. Weitere Informationen finden Sie unter [Verwenden der X-Ray-API mit der AWS CLI](#).

Verwenden Sie AWS X-Ray Service Maps und Trace Views mit API Gateway

In diesem Abschnitt finden Sie detaillierte Informationen zur Verwendung von [AWS X-Ray](#)-Service-Übersichten und Nachverfolgungsansichten mit API Gateway.

Themen

- [Beispiel für eine X-Ray-Service-Übersicht](#)
- [Beispiel für eine X-Ray-Ablaufverfolgungsansicht](#)

Beispiel für eine X-Ray-Service-Übersicht

AWS X-Ray Service Maps zeigen Informationen zu Ihrer API und all ihren nachgelagerten Diensten. Wenn X-Ray für eine API-Stufe in API Gateway aktiviert ist, wird in der Service-Übersicht ein Knoten angezeigt, der Informationen zur Gesamtzeit enthält, die im API Gateway-Service verbraucht wurde. Sie erhalten detaillierte Informationen über den Antwortstatus und ein Histogramm der API-Reaktionszeit für den ausgewählten Zeitraum. Für APIs die Integration mit AWS Diensten wie AWS Lambda Amazon DynamoDB werden Sie mehr Knoten sehen, die Leistungskennzahlen für diese Dienste bereitstellen. Es wird eine Service-Übersicht für jede API-Stufe geben.

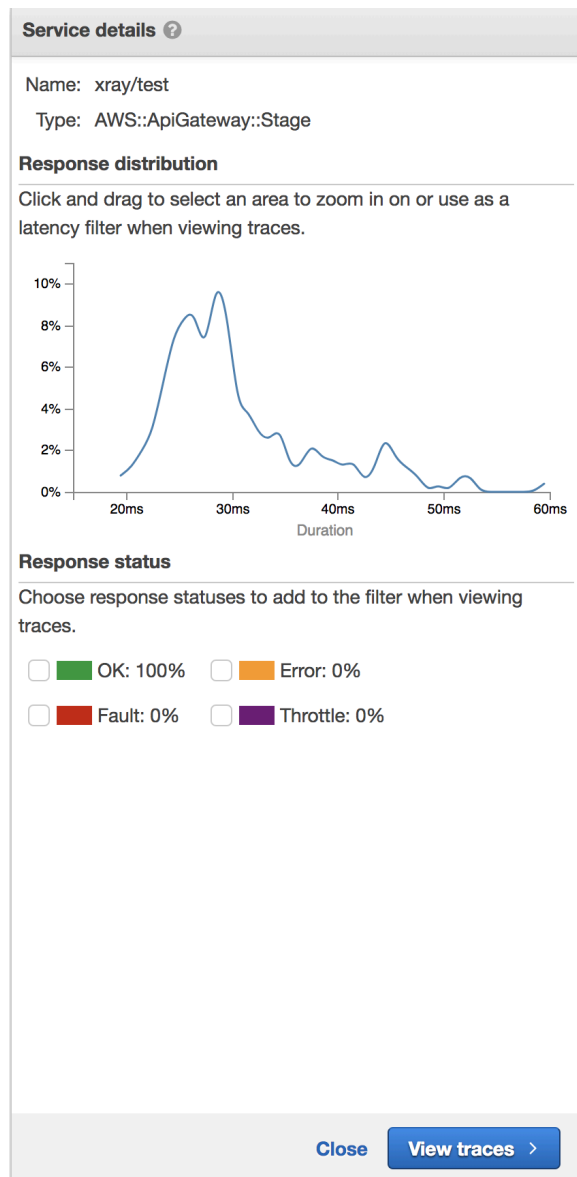
Das folgende Beispiel zeigt eine Service-Übersicht für die `test` Stufe einer aufgerufenen API `xray`. Diese API hat zwei Lambda-Integrationen. Die Knoten stellen den API-Gateway-Service und die beiden Lambda-Funktionen dar.

Eine detaillierte Erläuterung der Service-Übersichtsstruktur finden Sie unter [Verwenden der X-Ray-Trace-Map](#).



Sie können die Service-Übersicht vergrößern und eine Ansicht der Nachverfolgung für die API-Stufe anzeigen. Die Nachverfolgung zeigt detaillierte Informationen zu Ihrer API an, die als Segmente und Untersegmente dargestellt werden. Beispiel: Die Ablaufverfolgung für die Service-Übersicht umfasst die oben gezeigten Segmente für den Lambda-Service und die Lambda-Funktion. [Weitere Informationen finden Sie unter AWS Lambda und. AWS X-Ray](#)

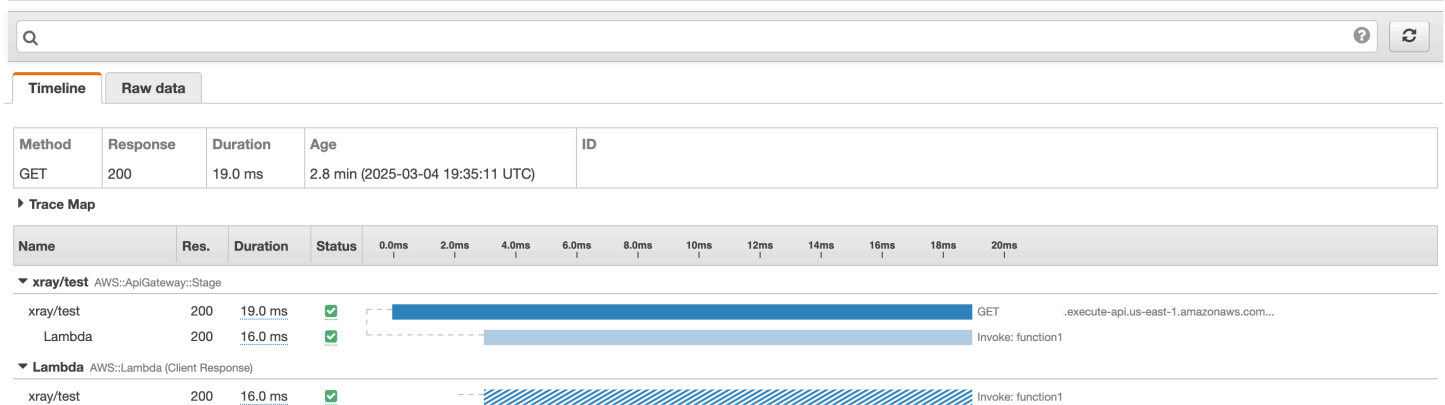
Wenn Sie einen Knoten oder ein Edge in einer X-Ray-Service-Übersicht auswählen, zeigt die X-Ray-Konsole ein Histogramm der Latenzverteilung. Sie können ein Latenzhistogramm verwenden, um zu sehen, wie lange es dauert, bis ein Service seine Anforderungen erfüllt. Es folgt ein Histogramm der API Gateway-Stufe, die in der vorherigen Service-Übersicht `xray/test` benannt wurde. Eine detaillierte Erklärung der Latenz-Verteilungshistogramme finden Sie unter [Verwenden von Latenzhistogrammen](#).



Beispiel für eine X-Ray-Ablaufverfolgungsansicht

Das folgende Diagramm zeigt eine Ablaufverfolgungsansicht, die für die oben beschriebene Beispiel-API mit einer Lambda-Backend-Funktion generiert wurde. Eine erfolgreiche API-Methodenanforderung wird mit einem Antwortcode von 200 angezeigt.

Eine detaillierte Erläuterung der Ablaufverfolgungsansichten finden Sie unter [Anzeigen von Ablaufverfolgungsdetails](#).

[Traces](#) > Details

AWS X-Ray Sampling-Regeln für API Gateway konfigurieren APIs

Sie können die AWS X-Ray Konsole oder das SDK verwenden, um Sampling-Regeln für Ihre Amazon API Gateway zu konfigurieren. Eine Samplingregel gibt an, welche Anforderungen X-Ray für Ihre API aufzeichnen soll. Durch das Anpassen von Samplingregeln können Sie die Menge der von Ihnen aufgezeichneten Daten steuern und das Stichprobenverhalten im laufenden Betrieb ändern, ohne Ihren Code ändern oder neu implementieren zu müssen.

Bevor Sie Ihre X-Ray-Samplingregeln festlegen, lesen Sie die folgenden Themen im X-Ray-Entwicklerhandbuch:

- [Konfigurieren von Samplingregeln](#)
- [Verwenden von Samplingregeln mit der X-Ray-API](#)

Themen

- [Optionswerte für Röntgenprobenabstastregeln für API Gateway APIs](#)
- [Beispiele für X-Ray-Samplingregeln](#)

Optionswerte für Röntgenprobenabstastregeln für API Gateway APIs

Die folgenden X-Ray-Samplingoptionen sind für API Gateway relevant. String-Werte können Platzhalter verwenden, um ein einzelnes Zeichen (?) Oder null oder mehr Zeichen (*) zu finden. Für weitere Informationen, einschließlich einer detaillierten Erklärung wie die Einstellungen Reservoir und Rate verwendet werden, finden Sie unter [Konfigurieren von Samplingregeln](#).

- **Regelname (Zeichenfolge)** – Ein eindeutiger Name für die Regel.

- **Priorität** (Ganzzahl zwischen 1 und 9999) – Die Priorität der Samplingregel. Services werten Regeln in aufsteigender Reihenfolge der Priorität aus und treffen eine Sampleentscheidung mit der ersten übereinstimmenden Regel.
- **Reservoir** (nicht negative Ganzzahl) – Eine feste Anzahl übereinstimmender Anfragen an das Gerät pro Sekunde vor Anwendung des festen Satzes. Das Reservoir wird nicht direkt von Services verwendet, sondern gilt für alle Services, die die Regel gemeinsam verwenden.
- **Rate** (Anzahl zwischen 0 und 100) – Der Prozentsatz der übereinstimmenden Anfragen an das Gerät, nachdem das Reservoir erschöpft ist.
- **Servicename** (Zeichenfolge) – Der API-Stufenname in Form von **{api-name}/{stage-name}**. Wenn Sie die [PetStore](#) Beispiel-API beispielsweise in einer Phase mit dem Namen bereitstellen würden, wäre der Wert für den Dienstenamen **test**, den Sie in Ihrer Stichprobenregel angeben müssen **pets/test**, wie folgt:
- **Servicetyp** (Zeichenfolge) – Für eine API Gateway-API kann entweder **AWS::ApiGateway::Stage** oder **AWS::ApiGateway::*** angegeben werden.
- **Host** (Zeichenfolge) – Der Hostname aus dem HTTP-Host-Header. Setzen Sie dies auf *****, um mit allen Hostnamen übereinzustimmen. Oder geben Sie einen vollständigen oder teilweisen Hostnamen an, z. B. **api.example.com** oder ***.example.com**.
- **Ressourcen-ARN** (Zeichenfolge) – Der ARN der API-Stufe, z. B. **arn:aws:apigateway:region::restapis/api-id/stages/stage-name**.

Der Stufenname kann von der Konsole oder der API Gateway-CLI oder -API abgerufen werden. Weitere Informationen zu ARN-Formaten finden Sie unter [Allgemeine Amazon Web Services-Referenz](#).

- **HTTP-Methode** (Zeichenfolge) – Die Methode, die gesampelt werden soll, z. B. **GET**.
- **URL path** (URL-Pfad) (Zeichenfolge) – Der URL-Pfad der Anforderung.
- (optional) **Attribute** (Schlüssel und Wert) – Header aus der ursprünglichen HTTP-Anforderung, z. B. **Connection**, **Content-Length** oder **Content-Type**. Jeder Attributwert kann bis zu 32 Zeichen lang sein.

Beispiele für X-Ray-Samplingregeln

Beispiel #1 für Samplingregeln

Diese Regel sampelt alle GET Anforderungen für die testxray API- test Stufe.

- **Rule name** (Regelname — **test-sampling**)

- Priorität — **17**
- Reservoirgröße — **10**
- Bestimmtes Zeitintervall — **10**
- Servicename — **testxray/test**
- Servicetyp – **AWS::ApiGateway::Stage**
- HTTP-Methode — **GET**
- Ressourcen-ARN – *
- Host — *

Beispiel #2 für Samplingregeln

Diese Regel sampelt alle Anforderungen für die testxray API in der prod-Stufe.

- Rule name (Regelname — **prod-sampling**
- Priorität — **478**
- Reservoirgröße — **1**
- Bestimmtes Zeitintervall — **60**
- Servicename — **testxray/prod**
- Servicetyp – **AWS::ApiGateway::Stage**
- HTTP-Methode — *
- Ressourcen-ARN – *
- Host — *
- Attribute — **{}**

AWS X-Ray Ablaufverfolgungen für Amazon API Gateway APIs

In diesem Abschnitt werden AWS X-Ray Trace-Segmente, Untersegmente und andere Trace-Felder für Amazon API Gateway APIs behandelt.

Bevor Sie diesen Abschnitt lesen, lesen Sie sich die folgenden Themen im X-Ray-Entwicklerhandbuch durch:

- [Verwenden Sie ein AWS-Managementkonsole](#)
- [X-Ray-Segmentdokumente](#)

- [Konzepte](#)

Themen

- [Beispiele für Ablaufverfolgungsobjekte für eine API Gateway-API](#)
- [Verstehen der Ablaufverfolgung](#)

Beispiele für Ablaufverfolgungsobjekte für eine API Gateway-API

In diesem Abschnitt werden einige der Objekte erläutert, die in einer Ablaufverfolgung für eine API Gateway-API angezeigt werden.

Anmerkungen

Anmerkungen können in Segmenten und Untersegmenten vorkommen. Sie werden als Filterausdrücke in Stichprobenregeln zum Filtern von Nachverfolgungen verwendet. Weitere Informationen finden Sie unter [Konfigurieren von Samplingregeln](#).

Im Folgenden sehen Sie ein `annotations`-Objekt, in dem eine API-Stufe durch die API-ID und den API-Stufennamen identifiziert wird:

```
"annotations": {  
  "aws:api_id": "a1b2c3d4e5",  
  "aws:api_stage": "dev"  
}
```

Weitere Informationen zu Anmerkungen finden Sie unter [X-Ray-Segmentdokumente](#) und wählen Sie dann X-Ray-Segmentdokumente, Anmerkungen aus.

AWS Ressourcendaten

Das `aws`-Objekt wird nur in Segmenten angezeigt. Im Folgenden finden Sie ein Beispiel für ein `aws` Objekt, das der Standard-Sampling-Regel entspricht. Eine ausführliche Erläuterung der Samplingregeln finden Sie unter [Konfigurieren von Samplingregeln](#).

```
"aws": {  
  "xray": {  
    "sampling_rule_name": "Default"  
  },  
  "api_gateway": {  
    "account_id": "123412341234",  
  }  
}
```

```

    "rest_api_id": "a1b2c3d4e5",
    "stage": "dev",
    "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
  }
}

```

Weitere Informationen zum aws-Objekt finden Sie unter [X-Ray-Segmentdokumente](#) und wählen Sie dann X-Ray-Segmentdokumente, AWS Ressourcendaten aus.

Verstehen der Ablaufverfolgung

Im Folgenden finden Sie ein Ablaufverfolgungssegment für eine API Gateway-Stufe. Eine detaillierte Beschreibung der Felder, aus denen das Ablaufverfolgungssegment besteht, finden Sie unter [X-Ray-Segmentdokumente](#).

```

{
  "Document": {
    "id": "a1b2c3d4a1b2c3d4",
    "name": "testxray/dev",
    "start_time": 1533928226.229,
    "end_time": 1533928226.614,
    "metadata": {
      "default": {
        "extended_request_id": "abcde12345abcde=",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
      }
    },
    "http": {
      "request": {
        "url": "https://example.com/dev?username=demo&message=hellofromdemo/",
        "method": "GET",
        "client_ip": "192.0.2.0",
        "x_forwarded_for": true
      },
      "response": {
        "status": 200,
        "content_length": 0
      }
    },
    "aws": {
      "xray": {
        "sampling_rule_name": "Default"
      }
    }
  }
}

```

```

    },
    "api_gateway": {
      "account_id": "123412341234",
      "rest_api_id": "a1b2c3d4e5",
      "stage": "dev",
      "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
  },
  "annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
  },
  "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
  "origin": "AWS::ApiGateway::Stage",
  "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/stages/dev",
  "subsegments": [
    {
      "id": "abcdefgh12345678",
      "name": "Lambda",
      "start_time": 1533928226.233,
      "end_time": 1533928226.6130002,
      "http": {
        "request": {
          "url": "https://example.com/2015-03-31/functions/arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
          "method": "GET"
        },
        "response": {
          "status": 200,
          "content_length": 62
        }
      },
      "aws": {
        "function_name": "xray123",
        "region": "us-east-1",
        "operation": "Invoke",
        "resource_names": [
          "xray123"
        ]
      }
    },
    {
      "namespace": "aws"
    }
  ]
}

```

```
    },  
    "Id": "a1b2c3d4a1b2c3d4"  
}
```


API-Gateway-Portale

Sie können ein API-Gateway-Portal verwenden, um Ihre APIs und Ihre API-Dokumentation auf einer Webseite mit API-Verbrauchern zu teilen. Ein Portal enthält eine Reihe von Portalprodukten. Jedes Portalprodukt ist eine logische Gruppierung von REST APIs und enthält die Dokumentation, die Sie für Ihre API-Nutzer erstellen und veröffentlichen. Produktseiten innerhalb eines Portals enthalten die benutzerdefinierte Dokumentation auf Portalproduktebene. Die REST-Endpunktseiten der Produkte enthalten die Dokumentation für jeden REST APIs sowie den Pfad, die Methode und die Phase einer REST-API. Die Kombination aus Produktseiten und REST-Endpunktseiten der Produkte bietet die vollständige Dokumentation, damit API-Nutzer lernen können, wie sie mit der Nutzung Ihres REST beginnen können APIs.

Verwenden Sie ein Portal, um Folgendes zu tun:

- Verwenden Sie API Gateway für Ihren gesamten API-Lebenszyklus, von der Erstellung Ihrer API über deren Dokumentation APIs bis hin zur Verteilung.
- Teilen Sie Portalprodukte für mehrere AWS Konten.
- Passen Sie einen zentralen Ort für Ihre Portalprodukte an, teilen Sie ihn und stellen Sie Produktseiten bereit.

Überlegungen

Die folgenden Überlegungen könnten sich auf Ihre Nutzung von API Gateway Gateway-Portalen auswirken:

- Sie können nur Portale erstellen, um REST gemeinsam zu nutzen APIs.
- Portale werden auf AWS Regionsebene erstellt. Ihr Portal kann nur REST enthalten APIs , die sich in derselben Region befinden, in der Sie das Portal erstellt haben.
- Sie können das AWS-Managementkonsole, oder nur verwenden AWS CLI, um Ihre Portale AWS SDKs zu erstellen und gemeinsam zu nutzen.
- Sie können den Zugriff auf Ihr Portal nur mithilfe von Amazon Cognito Cognito-Benutzerpools steuern. Ihre Portalnutzer können sich entweder direkt über einen Benutzerpool anmelden oder sich über einen externen Identitätsanbieter (IdP) zusammenschließen. Wenn Sie einen Amazon Cognito Cognito-Benutzerpool verwenden, können Sie kein geheimes Client-Geheimnis für Ihren Benutzerpool festlegen. Weitere Informationen finden Sie unter [Amazon Cognito Cognito-Benutzerpools](#).

Portalprodukte in API Gateway

Ein Portalprodukt steht für einen Service oder eine Funktionalität, die Sie teilen möchten. Ihr Portalprodukt ist eine Sammlung von Produkt-REST-Endpunkten und Produktseiten. Produkt-REST-Endpunkte sind die Zugriffspunkte zu Ihrem Portalprodukt. Sie bestehen aus dem Pfad und der Methode einer REST-API sowie der Phase, in der sie bereitgestellt wird. Bei Produktseiten handelt es sich um eine Dokumentation, die Sie bereitstellen, um zu erklären, wie API-Nutzer Ihre Produktendpunkte verwenden können. Ein Portalprodukt kann die gesamte Prod Phase als Sammlung von Produkt-REST-Endpunkten oder nur die GET /pets Ressource enthalten, die für die Prod Phase als einzelner Produkt-REST-Endpunkt bereitgestellt wurde.

Ihr Portalprodukt ist anpassbar. Sie können benutzerdefinierte Dokumentation hinzufügen, die REST-Endpunkte des Produkts umbenennen, die Anzeigereihenfolge neu organisieren, neue Abschnitte hinzufügen und Produkte kontenübergreifend AWS gemeinsam nutzen. Damit alle Änderungen, die Sie an Ihrem Portalprodukt vornehmen, wirksam werden, müssen Sie alle Portale, die Ihr Portalprodukt verwenden, erneut veröffentlichen.

Produktbeispiel für ein Portal zur Adoption von Haustieren

Als Beispiel könnten Sie mehrere REST haben, APIs die für einen Service zur Adoption von Haustieren stehen. Sie könnten API Gateway verwenden, um ein pet adoption Portalprodukt zu erstellen. Dieses Portalprodukt würde Kunden dabei helfen, herauszufinden, welches Produkt APIs sie verwenden sollten, um Haustiere kennenzulernen und zu adoptieren. Dieses Portalprodukt verwendet REST APIs , das Sie bereits erstellt haben, ermöglicht es Ihnen jedoch, sie neu zu gruppieren und zu organisieren. Sie können auch Unterlagen zu den Nutzungsbedingungen für Ihr Portalprodukt zur Adoption von Haustieren bereitstellen und Ihren Kunden die Möglichkeit geben, Ihr APIs Produkt auszuprobieren. Diese Informationen sind alle in Ihrem Portalprodukt gespeichert.

In der folgenden Tabelle sind drei Produkte aufgeführt APIs , die für ein Portalprodukt zur Adoption von Haustieren stehen, sowie die entsprechenden REST-Endpunkt-Operationsnamen und Seitenabschnittsnamen für das jeweilige Produkt.

RESTLICHE API-ID	REST-API-Pfad und -Methode	REST-API- Phase	Vorgangsname	Seitenabschnitt
kf5387miad	GET /dogs	Prod	View dogs	AdoptAnim als

RESTLICHE API-ID	REST-API-Pfad und -Methode	REST-API- Phase	Vorgangsname	Seitenabschnitt
kf5387miad	GET /dogs/ {dogId}	Prod	View dog	AdoptAnim als
ra8obxcevg	GET /cats	Prod	View cats	AdoptAnim als
ra8obxcevg	GET /cats/ {catId}	Prod	View cat	AdoptAnim als
h0rpx9cm62	ANY /user/ {userId}/ {petId+}	Beta	Request visit	AdoptProc ess

In diesem Beispiel sind zwei REST ra8obxcevg und APIkf5387miad, in dem AdoptAnim als Abschnitt gruppiert. Das Ergebnis dieser Bestellnavigation würde in einem Portal wie folgt aussehen:

The screenshot displays the Amazon API Gateway console interface for a 'Pet adoption portal'. The left sidebar contains a navigation menu with 'pet adoption' selected, showing 'Documentation' and 'API reference'. The 'API reference' section lists endpoints under 'AdoptAnimals' and 'AdoptProcess'. The main content area shows the 'View cat' endpoint details, including its URL, description, arguments, request body, and a 200 response example. Below this, the 'View cats' endpoint is partially visible. A footer note states 'Last update published on: November 17, 2025, 20:59 (UTC-08:00)'.

Pet adoption portal

Search

pet adoption

Documentation
Pet adoption

API reference

- ▼ **AdoptAnimals**
 - View cat (GET)
 - View cats (GET)
 - View dogs (GET)
 - View dog (GET)
- ▼ **AdoptProcess**
 - Request visit (GET)
 - Request visit (POST)
 - Request visit (PUT)
 - Request visit (PATCH)
 - Request visit (DELETE)
 - Request visit (OPTIONS)
 - Request visit (HEAD)

View cat

GET `https://ra8obxcvg.execute-api.us-west-1.amazonaws.com/Prod/cats/{catId}`

View a cat for adoption.

Arguments

- catId (string): The cat ID.
Required: Yes

Request body

The request does not have a request body.

Responses

200

```
1 {
2   "catId": "string",
3   "catAge": number
4 }
```

View cats

GET `https://ra8obxcvg.execute-api.us-west-1.amazonaws.com/Prod/cats`

View all the cats available for adoption.

Request body

The request does not have a request body.

Last update published on: November 17, 2025, 20:59 (UTC-08:00)

Da es sich bei der ANY Methode um eine Sammelmethode handelt, zeigt das Portal alle unterstützten HTTP-Methoden an. Dieses Portal enthält auch Dokumentation, die vom Produktbesitzer des Portals erstellt wurde.

Nächste Schritte

Um mit Portalprodukten zu beginnen, können Sie wie folgt vorgehen:

- Informationen zum Erstellen eines Portalprodukts finden Sie unter [the section called “Erstellen Sie ein Portalprodukt”](#).
- Weitere Informationen zur Funktion „Testen“ finden Sie unter [the section called “Aktivieren Sie Try it für einen Produkt-REST-Endpunkt in Ihrem Portal”](#).
- Weitere Informationen zu Produktseiten finden Sie unter [the section called “Erstellen Sie eine Produktseite”](#).

- Weitere Informationen zum Teilen Ihres Portalprodukts finden Sie unter [the section called “Portalprodukte teilen”](#).

Nachdem Sie ein Portalprodukt erstellt haben, können Sie es in einem Portal veröffentlichen. Weitere Informationen finden Sie unter [the section called “Erstellen Sie ein Portal”](#).

Erstellen Sie ein Portalprodukt in API Gateway

Das folgende Verfahren zeigt, wie Sie ein Portalprodukt erstellen. Ein Portal ist eine Sammlung von Portalprodukten. Nachdem Sie Ihr Portalprodukt erstellt haben, erstellen Sie Produkt-REST-Endpunkte und Produktseiten. Weitere Informationen zu Portalprodukten finden Sie unter [the section called “Portalprodukt”](#).

Überlegungen

Die folgenden Überlegungen können sich darauf auswirken, wie Sie ein Portalprodukt erstellen:

- Ihr Portalprodukt kann sowohl private als auch öffentliche REST-Dateien enthalten APIs. Private werden für die Funktion „Testen“ APIs nicht unterstützt und haben daher einen visuellen Unterschied in Ihrem Portal. Als Portalbesitzer müssen Sie möglicherweise eine Dokumentation vorlegen, um dies zu erläutern.
- Wenn Sie Ihr Portalprodukt mit dem AWS CLI oder erstellen AWS SDKs, hat Ihr Portal keine Produktendpunkte oder Produktseiten. Sie müssen diese Ressourcen mithilfe der AWS CLI OR-Konsole hinzufügen. Informationen zum Erstellen eines REST-Endpunkts für ein Produkt finden Sie unter [the section called “Erstellen Sie einen Produkt-REST-Endpunkt”](#). Informationen zum Erstellen einer Produktseite finden Sie unter [the section called “Erstellen Sie eine Produktseite”](#).

Erstellen Sie ein Portalprodukt

Das folgende Verfahren zeigt, wie Sie ein Portalprodukt erstellen.

Um ein Portalprodukt zu erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie Produkt erstellen aus.

4. Geben Sie unter Produktname den Namen Ihres Portalprodukts ein.
5. Geben Sie für Produktbeschreibung eine Beschreibung ein.
6. Wählen Sie Weiter aus.
7. Um Ihre Produkt-REST-Endpunkte auszuwählen, wählen Sie unter API-Endpunkte eine API und dann eine Phase aus.
8. Um Ihren Produkt-REST-Endpunkten einen Endpunkt hinzuzufügen, wählen Sie den API-Endpunkt aus und wählen Sie dann Zum Produkt hinzufügen aus.

Note

Wählen Sie nicht Weiter, ohne zuerst „Zum Produkt hinzufügen“ auszuwählen.

The screenshot shows the 'Manage endpoints' page in the Amazon API Gateway console. On the left, under 'API endpoints (2)', there is a list of endpoints: 'API endpoints', 'GET /dogs', and 'GET /dogs/(dogid)'. The 'GET /dogs' endpoint is selected. A red box highlights the 'Add to product' button. On the right, under 'Selected API endpoints (3)', there is a list of endpoints: 'API endpoints', 'GET /cats', 'GET /cats/(catID)', and 'ANY /user/(userId)/(petid+)'. The 'GET /cats' endpoint is selected. At the bottom right, there are 'Cancel' and 'Submit' buttons.

Der API-Endpunkt wird in der Liste Ausgewählte API-Endpunkte angezeigt.

9. Wählen Sie Weiter aus.
10. Überprüfen Sie Ihre Auswahl und wählen Sie Produkt erstellen.

Nachdem Sie Ihr Portalprodukt mithilfe der Konsole erstellt haben, handelt es sich bei all Ihren Produktseiten und REST-Endpunktseiten um Entwürfe, die nicht in einem Portal angezeigt werden. Damit Ihre Produktseiten und REST-Endpunktseiten für Verbraucher sichtbar sind, müssen Sie Ihren Entwurf zu einem Abschnitt hinzufügen. Wenn Sie Ihr Portalprodukt mithilfe von AWS CLI oder erstellen AWS SDKs, fügen Sie den Entwurf dem Abschnitt im AWS CLI Befehl hinzu. Unabhängig davon, wie Sie Ihren Entwurf zu einem Abschnitt hinzufügen, müssen Sie das Portal veröffentlichen, das Ihre Portalprodukte verwendet, damit er für Verbraucher sichtbar ist.

Um Ihren Entwurf zu einem Seitenabschnitt hinzuzufügen

1. Ihre Entwürfe werden auf der Registerkarte Dokumentation aufgeführt. Es gibt Dokumentationsentwürfe für Ihre Produktseiten und API-Entwurfsseiten für Ihre REST-Endpunktseiten für Ihre Produkt-REST-Endpunktseiten. Wählen Sie API-Referenzseiten für Entwürfe aus.
2. Wählen Sie einen Entwurf einer API-Referenzseite aus.

Wenn Sie keine REST-Endpunkte für Produkte haben, werden Sie auch keine API-Referenzseiten für Entwürfe haben. Informationen zum Erstellen eines REST-Endpunkts für Produkte finden Sie unter [the section called “Erstellen Sie einen Produkt-REST-Endpunkt”](#).

3. Wählen Sie „Seite bearbeiten“.
4. Auf dieser Seite können Sie alle vorhandenen API-Dokumentationsteile überschreiben oder die API Gateway Gateway-Dokumentation verwenden. Damit der Inhalt der REST-Endpunktseite Ihres Produkts für Verbraucher sichtbar ist, geben Sie unter Abschnittsname einen Namen ein. Wenn dies der `/pets-GET` Endpunkt wäre, könnte es der Seitenname sein `Pets`.
5. Wählen Sie Änderungen speichern aus.
6. Der neue Seitenname, den Sie erstellt haben, wird im Abschnitt API-Referenzseiten angezeigt.

Damit die neue Seite für Verbraucher sichtbar ist, müssen Sie Ihr Portal dennoch erneut veröffentlichen. Weitere Informationen finden Sie unter [the section called “Veröffentlichen Sie ein Portal”](#).

Erstellen Sie einen Produkt-REST-Endpunkt in API Gateway

Ein Produkt-REST-Endpunkt ist ein Zugriffspunkt zu Ihrem Portalprodukt. Jeder Endpunkt besteht aus dem Pfad und der Methode einer REST-API sowie der Phase, in der sie bereitgestellt wird. Der Endpunkt enthält jegliche Dokumentation, die Sie für Ihre REST-API definiert haben, oder kann mit benutzerdefinierter Dokumentation aktualisiert werden. Wenn Sie Ihren Produkt-REST-Endpunkt erstellen, konfigurieren Sie die folgenden Einstellungen:

- Der Betriebsname des REST-Endpunkts des Produkts. Wenn Ihre REST-API den Pfad und die Methode hat `GET /pets`, lautet der Name in einem Portal `/pets`. Mithilfe des Vorgangsnamens können Sie den Namen des REST-Endpunkts des Produkts in ändern `Pet summaries`. Dies hat keine Auswirkungen auf Ihre bestehende API.

- Die Try It-Funktionalität für Ihren Produktendpunkt. Auf diese Weise können API-Nutzer Ihre App APIs in Ihrem Portal testen. Wenn Sie einen Produktendpunkt für eine private API erstellen, ist die Testfunktion auf eingestellt UNAVAILABLE. Weitere Informationen finden Sie unter [the section called “Aktivieren Sie Try it für einen Produkt-REST-Endpunkt in Ihrem Portal”](#).
- Wenn Sie derzeit Dokumentationsteile für Ihre REST-API definiert haben, können Sie entweder die Dokumentation importieren oder Ihre bestehende Dokumentation überschreiben und mit [GitHub Flavored](#) Markdown eine neue Dokumentation auf Produktebene erstellen. Alle Dokumentationsüberschreibungen werden nicht auf die Teile Ihrer API Gateway Gateway-API-Dokumentation zurückübertragen. Wenn Sie Ihre aktuellen Dokumentationsteile importieren, synchronisiert API Gateway Änderungen in Ihrer API mit Ihrem Produkt-REST-Endpunkt.

Erstellen Sie einen Produkt-REST-Endpunkt

Das folgende Verfahren zeigt, wie Sie einem vorhandenen Portalprodukt einen Produkt-REST-Endpunkt hinzufügen. Informationen zum gemeinsamen Erstellen eines Portalprodukts und eines Produkt-REST-Endpunkts finden Sie unter [the section called “Erstellen Sie ein Portalprodukt”](#).

So erstellen Sie einen Produkt-REST-Endpunkt für ein Produktportal

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie ein Portalprodukt aus.
4. Wählen Sie die Registerkarte Zugeordnete Endpunkte und anschließend Endpunkte hinzufügen aus.
5. Um Ihre Produkt-REST-Endpunkte auszuwählen, wählen Sie unter API-Endpunkte eine API und dann eine Phase aus.
6. Um Ihren Produkt-REST-Endpunkten einen Endpunkt hinzuzufügen, wählen Sie den API-Endpunkt aus und wählen Sie dann Zum Produkt hinzufügen aus.

Der API-Endpunkt wird in der Liste Ausgewählte API-Endpunkte angezeigt.

7. Wählen Sie Absenden aus.
8. Ihre REST-Endpunktseite für Ihr Produkt ist jetzt ein Entwurf. Wählen Sie „API-Referenzseiten entwerfen“ und anschließend Ihre REST-Endpunktseite für Ihr Produkt aus, um den Inhalt der Seite fertigzustellen.
9. Wählen Sie „Seite bearbeiten“.

10. Um die bestehende API Gateway Gateway-Dokumentation zu überschreiben, aktivieren Sie für Dokumentationsquelle die Option Dokumentation überschreiben und gehen Sie wie folgt vor:
 - a. Wählen Sie für Inhalt anzeigen die Option Überschreiben erstellen aus.
 - b. Geben Sie unter Operationsname einen neuen Operationsnamen ein.
 - c. Geben Sie für Seitentext Ihre benutzerdefinierte Dokumentation ein.
 - d. (Optional) Geben Sie für Endpoint einen benutzerdefinierten Endpunkt ein, der in Ihrem Portal angezeigt wird.
11. Wählen Sie Funktionalität testen aus, damit Kunden Ihre API in Ihrem Portal aufrufen können. Weitere Informationen finden Sie unter [the section called “Aktivieren Sie Try it für einen Produkt-REST-Endpunkt in Ihrem Portal”](#).
12. Geben Sie als Abschnittsname einen Namen ein oder wählen Sie einen vorhandenen Abschnitt aus. Wenn Sie Ihren Produkt-REST-Endpunkt zu einem Abschnitt hinzufügen, können Verbraucher Ihre Inhalte auf einem veröffentlichten Portal einsehen.
13. Wählen Sie Änderungen speichern aus.

Damit die neue Seite für Verbraucher sichtbar ist, müssen Sie Ihr Portal dennoch erneut veröffentlichen. Weitere Informationen finden Sie unter [the section called “Veröffentlichen Sie ein Portal”](#).

Erstellen Sie eine Produktseite in API Gateway

Eine Produktseite hilft Ihren Produktkonsumenten, Ihre Produkte zu verstehen und zu verwenden. Die Dokumentation bezieht sich auf Produktebene, sodass alle Portale, die Ihr Produkt anbieten, über diese Dokumentation verfügen. API Gateway unterstützt zwei Arten von Produktseiten, die in [GitHub Flavored Markdown](#) geschrieben wurden:

Seite „Übersicht“

Wenn Sie Ihr Produkt erstellen, generiert API Gateway automatisch eine Übersichtsseite für Ihr Produkt auf der Grundlage der Informationen, die Sie bei der Erstellung Ihres Produkts angeben. Diese Seite hat zusätzliche Formatierungen, die nicht geändert werden können. Sie können den Text ändern oder die Seite aus dem Portalprodukt entfernen.

Benutzerdefinierte Seiten

Eine benutzerdefinierte Seite ist eine in Markdown geschriebene Seite. Sie können diesen Abschnitt für die Allgemeinen Geschäftsbedingungen oder ein Tutorial zur Verwendung

Ihres Produkts verwenden. Sie können jede benutzerdefinierte Seite in eine Übersichtsseite umwandeln. Wenn Sie eine benutzerdefinierte Seite in eine Übersichtsseite ändern, wird der Seiteninhalt so formatiert, dass er der Übersichtsseite entspricht.

Überlegungen

Die folgenden Überlegungen könnten sich auf Ihre Verwendung von Produktseiten auswirken:

- Diese Dokumentation bezieht sich nicht auf API-Ebene und enthält keine Methoden, Ressourcen oder Anforderungsparameter. Informationen zum Ändern der Dokumentation auf API-Ebene für ein Produkt finden Sie unter [the section called “Aktualisieren Sie einen Produkt-REST-Endpunkt”](#)
- Sie können eine .md Datei nicht direkt in Ihr Produkt hochladen.
- Sie können die Reihenfolge der Produktseiten ändern, indem Sie die Seitenliste in der Anzeigereihenfolge Ihres Portalprodukts ändern. Weitere Informationen finden Sie unter [the section called “Aktualisieren Sie ein Portalprodukt”](#).
- Sie müssen alle Portale, die Ihr Produkt enthalten, erneut veröffentlichen, damit die Änderungen an der Dokumentation wirksam werden.
- Wenn Sie mehrere Portale haben, die dasselbe Portalprodukt verwenden, sind die Produktseiten auf allen Portalen identisch. Um dieselben Produkt-REST-Endpunkte, aber unterschiedliche Produktseiten zu haben, erstellen Sie mehrere Produkte.

Erstellen Sie eine Produktseite

Das folgende Verfahren zeigt, wie Sie eine Produktseite erstellen.

Um eine Produktseite zu erstellen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie ein Produkt aus.
4. Wählen Sie unter Dokumentation die Option Benutzerdefinierte Seite erstellen aus.
5. Geben Sie unter Seitentitel den Seitentitel ein.
6. Geben Sie für Body Ihre ergänzende Dokumentation mit GitHub Flavored Markdown ein.

Im Bereich Vorschau wird gezeigt, wie Ihre Inhalte in einem Portal angezeigt werden. Der endgültige visuelle Stil kann sich je nach Ihren Portaleinstellungen ändern.

7. Wählen Sie Produktseite erstellen aus.
8. Ihre Produktseite ist jetzt ein Entwurf. Wählen Sie Dokumentationsseiten entwerfen und wählen Sie dann Ihre Produktseite aus, um den Inhalt der Seite fertigzustellen.
9. Wählen Sie „Seite bearbeiten“.
10. Geben Sie als Abschnittsname einen Namen ein oder wählen Sie einen vorhandenen Abschnitt aus. Wenn Sie Ihre Produktseite zu einem Bereich hinzufügen, können Verbraucher Ihre Inhalte auf einem veröffentlichten Portal einsehen.
11. Wählen Sie Änderungen speichern aus.

Damit die neue Seite für Verbraucher sichtbar ist, müssen Sie Ihr Portal dennoch erneut veröffentlichen. Weitere Informationen finden Sie unter [the section called “Veröffentlichen Sie ein Portal”](#).

Aktualisieren Sie ein Portalprodukt in API Gateway

Wenn Sie ein Portalprodukt aktualisieren, können Sie den Namen, die Beschreibung oder die Anzeigereihenfolge ändern, in der Ihre Produkt-REST-Endpunkte und Produktseiten angezeigt werden. Um die Anzeigereihenfolge zu ändern, ändern Sie die Seitenliste für Produktseiten und den Abschnittsnamen für Produktendpunkte. Wenn Sie dies mit AWS CLI oder tun SDKs, geben Sie den ARN der Produktseite oder den Produkt-REST-Endpunkt ARN an. Sie müssen alle Portale, die Ihre Produkte verwenden, erneut für Ihre API-Nutzer veröffentlichen, damit die Änderungen in einem Portal widergespiegelt werden.

Aktualisieren Sie ein Portalprodukt

Das folgende Verfahren zeigt, wie Sie die Abschnittsliste ändern, um die REST-Endpunkte des Produkts so anzuordnen, wie sie in einem Portal angezeigt werden. In diesem Verfahren gehen wir davon aus, dass Ihr Portalprodukt mindestens zwei Produkt-REST-Endpunkte enthält.

Um ein Portalprodukt zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.

3. Wählen Sie ein Produkt aus.
4. Wählen Sie auf der Registerkarte Dokumentation für Produktseiten die Option Verwalten aus.
5. Ziehen Sie die REST-Endpunkte des Produkts per Drag-and-Drop, um sie neu anzuordnen. Sie können REST-Endpunkte und Produktseiten auch in den Bereich Entwurfsseiten verschieben. Die Seiten in den Entwurfsseiten werden in Ihrem Portal nicht sichtbar sein.
6. (Optional) Benennen Sie die REST-Endpunktnamen des Produkts um oder fügen Sie einen neuen API-Referenzabschnitt hinzu. Diese Änderungen wirken sich nicht auf Ihren REST aus APIs.
7. Wählen Sie Änderungen speichern aus.

Aktualisieren Sie einen Produkt-REST-Endpunkt in API Gateway

Wenn Sie einen Produkt-REST-Endpunkt aktualisieren, können Sie alle Einstellungen des Produktendpunkts ändern, mit Ausnahme des Pfads und der Methode einer REST-API sowie der Phase, in der sie bereitgestellt wird. Alle Änderungen an Ihren Produktendpunkten werden automatisch aktualisiert. Sie müssen jedoch alle Portale, die Ihre Produkte verwenden, erneut veröffentlichen, damit sich die Änderungen in einem Portal widerspiegeln.

Aktualisieren Sie einen Produkt-REST-Endpunkt

Das folgende Verfahren zeigt, wie Sie einen Produkt-REST-Endpunkt aktualisieren, um die aktuelle API-Dokumentation mit der benutzerdefinierten API-Referenzdokumentation zu überschreiben.

Um einen Produkt-REST-Endpunkt zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie ein Produkt aus.
4. Wählen Sie auf der Registerkarte Dokumentation unter API-Referenzseiten den Namen eines Produkt-REST-Endpunkts aus, z. B. /dogs - GET.
5. Wählen Sie für Vorschau die Option Seite bearbeiten aus.
6. Aktivieren Sie unter Dokumentationsquelle die Option Dokumentation überschreiben.
7. Wählen Sie für Inhalt anzeigen die Option Vorhandenen Inhalt überschreiben aus.

Wenn Sie „Gesamten Inhalt entfernen“ wählen, wird der Inhalt von der Seite entfernt. Sie können jedoch „Vorhandenen Inhalt überschreiben“ wählen, um erneut auf die Dokumentation zuzugreifen.

8. Geben Sie im Hauptteil der Seite die benutzerdefinierte API-Referenzdokumentation mit [GitHub Flavored Markdown](#) ein.

Diese Dokumentation wird nicht zurück an API Gateway weitergegeben.

9. Wählen Sie Änderungen speichern aus.
10. Sie müssen alle Portale, die dieses Portalprodukt verwenden, erneut veröffentlichen, damit die Änderungen wirksam werden.

Aktualisieren Sie eine Produktseite in API Gateway

Wenn Sie eine Produktseite aktualisieren, können Sie alle Einstellungen der Produktseite ändern. Alle Änderungen an Ihrer Produktseite werden automatisch aktualisiert. Sie müssen jedoch alle Portale, die Ihre Produkte verwenden, erneut veröffentlichen, damit sich die Änderungen in einem Portal widerspiegeln.

Portalprodukte in API Gateway teilen

Sie können ein Portalprodukt gemeinsam AWS-Konten nutzen AWS RAM. Wenn Sie ein Portalprodukt teilen, kann das andere Konto Ihr Portalprodukt in seinem eigenen Portal verwenden. Mit gemeinsam genutzten Portalprodukten können Sie einen einzigen Katalog Ihrer Organisation erstellen APIs und Governance-Standards in Ihrem gesamten API-Ökosystem durchsetzen. Gleichzeitig bietet die gemeinsame Nutzung von Portalprodukten API-Anbietern die Flexibilität, sie APIs in ihren eigenen Konten zu entwickeln, zu testen und zu verwalten.

Überlegungen

Die folgenden Überlegungen können sich darauf auswirken, wie Sie Portalressourcen gemeinsam nutzen:

- Wenn Sie Ihr Produkt mit einem anderen Konto teilen, kann dieses Konto keine Eigenschaften Ihrer REST-API ändern. Dazu gehören die Integrationsendpunkte, die Autorisierungsstrategie oder die Stufenkonfiguration.

- Wenn Sie das Portalprodukt eines anderen Kontos zu Ihrem Portal hinzufügen, kann der Eigentümer des Portalprodukts keine anderen Eigenschaften Ihres Portals anzeigen oder steuern. Der Portalprodukteigentümer weiß nur, dass das Produkt in Ihrem Portal verwendet wird.
- API Gateway Gateway-Portalprodukte werden auf der AWS-Region Ebene gemeinsam genutzt.
- Sie können eine einzelne Ressourcenfreigabe für mehrere Prinzipale verwenden und weitere Prinzipale hinzufügen, nachdem Sie die Ressourcenfreigabe erstellt haben. Wir empfehlen, dass Sie Ihre Ressourcenfreigabe nach Möglichkeit wiederverwenden.
- Wenn sich beide Konten in derselben Organisation befinden, die verwendet AWS Organizations, wird die gemeinsame Nutzung der Ressource automatisch akzeptiert. Sie müssen jedoch trotzdem mithilfe von AWS RAM eine Ressourcenfreigabe erstellen.
- Wenn sich beide Konten in derselben Organisation befinden AWS Organizations und die gemeinsame Nutzung von Ressourcen innerhalb Ihrer Organisation aktiviert ist, wird allen Prinzipalen in der Organisation, für die Sie die Ressourcen gemeinsam nutzen, automatisch Zugriff auf die Ressourcenfreigaben gewährt. Es ist keine Einladung erforderlich und Sie können die gemeinsame Ressourcennutzung überspringen.
- Wenn das Konto, mit dem Sie das Produkt geteilt haben, die gemeinsame Nutzung der Ressource nicht innerhalb von 12 Stunden akzeptiert, müssen Sie die Ressource erneut teilen.
- Nachdem Sie die Ressourcenfreigabe erstellt haben, AWS RAM aktualisiert die Produktfreigaberichtlinie Ihres Produkts, um den Zugriff auf Prinzipale ohne ausdrücklichen allow Zugriff zu verhindern. Weitere Informationen finden Sie unter [Ermitteln, ob eine Anforderung innerhalb eines Kontos zugelassen oder verweigert wird](#) im IAM-Benutzerhandbuch.

Die aktualisierte Ressourcenrichtlinie wird wie folgt aussehen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GetProduct",
        "apigateway:ListEndpoints",
        "apigateway:ListPages",
        "apigateway:GetEndpoint",
        "apigateway:GetPage"
      ],
      "Resource": [
```

```

        "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id",
        "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id/*"
    ]
}
]
}

```

AWS RAM hat verhindert, dass Principals ohne ausdrücklichen Zugriff Ihr Produkt zu ihrem Portal hinzufügen können. Dazu wurde Folgendes hinzugefügt:

```
"StringNotEquals": { "aws:PrincipalAccount": "555555555555" }
```

Informationen darüber, wie Sie ein Produkt teilen können, finden Sie unter [the section called “Teilen Sie Ihr Portalprodukt mit einem Portalbesitzer”](#). Informationen zum Hinzufügen eines für Sie geteilten Produkts zu Ihrem Portal finden Sie unter [the section called “Fügen Sie Ihrem Portal ein gemeinsam genutztes Portalprodukt hinzu”](#).

Teilen Sie Ihr Portalprodukt mit einem Portalbesitzer in API Gateway

Als Product Owner können Sie Ihr Produkt mit einem Portalbesitzer in einer anderen AWS-Konto Nutzung teilen AWS RAM. Sie behalten die volle Kontrolle über alle Produktseiten und Produkt-REST-Endpunktseiten.

Überlegungen

Die folgenden Überlegungen können sich darauf auswirken, wie Sie Portalprodukte gemeinsam nutzen:

- Sie können Ihre Portalprodukte jederzeit teilen oder deren Freigabe aufheben. Wenn Sie die Freigabe eines Portalprodukts aufheben, während es in einem veröffentlichten Portal verwendet wird, kann der Portalbesitzer das Portalprodukt nicht ansehen, alle daran vorgenommenen Aktualisierungen anzeigen oder das Portal erneut veröffentlichen oder eine Vorschau des Portals anzeigen, bis er das Portalprodukt entfernt hat.
- Sie können die Portalprodukte anzeigen, die Sie für andere Konten freigegeben haben.
- Sie können keine Portale ändern, die Ihr Portalprodukt enthalten, es sei denn, Sie erstellen das Portal selbst.

Teilen Sie Ihr Portalprodukt

Das folgende Verfahren zeigt, wie Sie eine Ressourcenfreigabe erstellen.

AWS-Managementkonsole

Informationen zur AWS-Managementkonsole Verwendung [von finden Sie unter Erstellen einer Ressourcenfreigabe AWS RAM im AWS RAM Benutzerhandbuch](#).

Wählen Sie unter Ressourcentyp auswählen die Option API Gateway Product aus.

AWS CLI

Im Folgenden [create-resource-share](#) wird eine Ressourcenfreigabe für Ihren privaten benutzerdefinierten Domainnamen erstellt. Es kann einige Minuten dauern, bis die Ressourcen- und Prinzipal-Zuordnungen abgeschlossen ist. Geben Sie für Prinzipale eine Konto-ID oder eine Organisations-ID an (z. B. `arn:aws:organizations::123456789012:organization/o-1234abcd`). Sie können mehrere Prinzipale für Ihre Ressourcenfreigabe angeben.

```
aws ram create-resource-share \  
  --region us-west-2 \  
  --name portal-product-resource-share \  
  --permission-arns arn:aws:ram::aws:permission/  
AWSRAMDefaultPermissionAPIGatewayDeveloperPortalProduct \  
  --resource-arns arn:aws:apigateway:us-west-2:111122223333:/portalproducts/  
p0000000000 \  
  --principals 222222222222
```

Um die Freigabe Ihres Portalprodukts aufzuheben, verwenden Sie AWS RAM zum Löschen der Ressourcenfreigabe.

Sie können die Richtlinie zur gemeinsamen Nutzung von Produkten jederzeit ändern, um festzulegen, welche Prinzipale Ihre Portalprodukte in ihren Portalen verwenden können.

Um die Richtlinie zur gemeinsamen Nutzung von Produkten zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie ein Produkt aus.

4. Wählen Sie auf der Registerkarte Produktfreigabe für Richtlinie zur gemeinsamen Nutzung von Produkten die Option Richtlinie zur gemeinsamen Nutzung von Produkten verwalten aus.
5. Wenn Sie Ihr Produkt noch nicht geteilt haben, verweigert Ihre Richtlinie zur Produktfreigabe jeglichen Zugriff auf andere Konten für Ihre Portalprodukte. Sie müssen die Richtlinie aktualisieren, um bestimmten Konten den Zugriff auf Ihr Portalprodukt zu ermöglichen.
6. Nachdem Sie Ihre Richtlinie zur gemeinsamen Nutzung von Produkten aktualisiert haben, wählen Sie Änderungen speichern aus.

Fügen Sie Ihrem Portal in API Gateway ein gemeinsam genutztes Portalprodukt hinzu

Als Portalbesitzer können Sie Portalprodukte verwenden, die von anderen Konten in Ihrem Portal für Sie freigegeben wurden. Sie behalten die volle Kontrolle über Ihr Portal.

Überlegungen

Die folgenden Überlegungen können sich darauf auswirken, wie Sie Portalressourcen gemeinsam nutzen:

- Sie müssen die Freigabe von Produktressourcen vom Product Owner akzeptieren, um sein Produkt in Ihrem Portal verwenden zu können. Wenn Sie und der Product Owner derselben Organizations angehören, AWS RAM kann er einige Schritte zum Teilen für Sie durchführen.
- Wenn der Product Owner die gemeinsame Nutzung eines Produkts rückgängig macht, während es sich in Ihrem veröffentlichten Portal befindet, ist das Produkt weiterhin im Portal sichtbar, aber Sie können das Portal erst wieder veröffentlichen, wenn Sie dieses Produkt entfernen. Es liegt in Ihrer Verantwortung als Portalbesitzer, das Produkt aus Ihrem Portal zu entfernen, da der Product Owner das nicht für Sie tun kann.
- Sie können die Portalprodukte, die Produktseiten und die REST-Endpunktseiten der Produkte anzeigen, aber Sie können keine dieser Ressourcen ändern.
- Wenn ein Produkt mit Ihnen geteilt wird, können Sie es nicht mit einem anderen Konto teilen.

(Optional) Akzeptieren Sie die gemeinsame Nutzung der Ressource

Nachdem Ihr Product Owner eine Ressourcenfreigabe erstellt hat, haben Sie 12 Stunden Zeit, diese zu akzeptieren. Wenn Sie derselben Organisation angehören, die AWS Organizations auch der Product Owner verwendet, wird die Freigabe automatisch akzeptiert. Wenn Sie einer Organisation angehören, in der die automatische Ressourcenfreigabe aktiviert ist, wird die Ressource automatisch für Sie freigegeben.

AWS-Managementkonsole

Informationen zur Verwendung von finden Sie im AWS-ManagementkonsoleAWS RAM Benutzerhandbuch unter [Annehmen und Ablehnen von Einladungen zur gemeinsamen Nutzung von Ressourcen](#).

AWS CLI

Verwenden Sie den folgenden [get-resource-share-invitations](#)Befehl, um alle Ressourcen zu finden, die mit Ihnen geteilt wurden:

```
aws ram get-resource-share-invitations \
  --region us-west-2
```

Verwenden Sie den resultierenden Ressourcenfreigabe-ARN, um die Einladung zur Ressourcenfreigabe anzunehmen. Der folgende [accept-resource-share-invitation](#)Befehl akzeptiert die gemeinsame Nutzung der Ressource.

```
aws ram accept-resource-share-invitation \
  --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
  --region us-west-2
```

Fügen Sie Ihrem Portal ein gemeinsam genutztes Produkt hinzu

Nachdem Sie die gemeinsame Nutzung der Ressource akzeptiert haben, fügen Sie das Produkt zu Ihrem Portal hinzu.

AWS-Managementkonsole

Um Ihrem Portal ein gemeinsam genutztes Produkt hinzuzufügen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wählen Sie auf der Registerkarte Produkte für Portal-Produkte die Option Produkte hinzufügen aus.

5. Fügen Sie Ihrem Portal ein Produkt hinzu. Geteilte Produkte werden wie shared in der Produktliste angezeigt.
6. Wählen Sie Änderungen speichern aus.

AWS CLI

Verwenden Sie den folgenden `get-portal-products` Befehl, um alle Portalprodukte zu finden, die für Sie freigegeben wurden:

```
aws apigatewayv2 get-portal-products \
  --resource-owner OTHER_ACCOUNTS \
  --region us-west-2
```

Verwenden Sie den folgenden `update-portal` Befehl, um Ihrem Portal ein gemeinsam genutztes Portalprodukt hinzuzufügen:

```
aws apigateway update-portal \
  --included-portal-product-arns arn:aws:apigateway:us-west-2:111122223333:/
portalproducts/p0000000000 \
  --region us-west-2
```

Aktivieren Sie Try it für einen REST-Endpunkt eines API Gateway-Produkts in Ihrem Portal

Verwenden Sie try it, damit ein API-Nutzer Ihren Produktendpunkt von Ihrem Portal aus aufrufen kann. Wenn ein API-Nutzer try it verwendet, gibt er Methodenanforderungsparameter ein und ruft Ihren Produktendpunkt auf. Dann ruft API Gateway Ihre API über das öffentliche Internet auf und gibt die Antwort im Testfenster zurück. Sie können eine benutzerdefinierte Produktseite verwenden, um zusätzliche Informationen zu den erforderlichen Parametern für den Zugriff auf Ihre API bereitzustellen.

API Gateway verwendet die folgenden Grenzwerte, um Ihre zu schützen APIs:

- API Gateway erlaubt nur 3 Anfragen pro Sekunde an Ihre API.
- API Gateway verwendet ein integriertes Timeout-Limit von 29000 ms. Ihre aktuelle API hat möglicherweise ein höheres Timeout-Limit, aber API Gateway wendet dieses Timeout nicht an, wenn ein Kunde es versucht.

- API Gateway begrenzt die Antwortnutzlast auf 6 MB.

Überlegungen

Die folgenden Überlegungen können sich auf die Art und Weise auswirken, wie Sie es ausprobieren:

- Try it wird in der Vorschau eines Portals nicht unterstützt.
- Try it wird für den REST APIs mit den folgenden Funktionen nicht unterstützt:
 - Privat APIs
 - APIs die gegenseitiges TLS verwenden
 - APIs die private oder selbstsignierte SSL-Zertifikate verwenden

Als Portalbesitzer sind Sie dafür verantwortlich, mit Ihren API-Nutzern zu kommunizieren. Die Begründung für die Schaltfläche „Warum versuchen“ gibt es nicht für REST APIs , die nicht unterstützt werden. API Gateway erklärt Ihnen das nicht.

Aktivieren Sie „Testen Sie es“ für einen REST-Endpunkt eines Produkts

Das folgende Verfahren zeigt, wie Sie Try It für einen Produktendpunkt aktivieren.

Informationen zur Verwendung von Try-it in einem Portal finden Sie unter [the section called „Verwenden Sie ein Portal“](#).

Aktivieren Sie Try it für einen Produkt-REST-Endpunkt

1. Wählen Sie im Hauptnavigationsbereich die Option Portal-Produkte aus.
2. Wählen Sie ein Produkt aus.
3. Wählen Sie auf der Registerkarte Dokumentation unter API-Referenzseiten den Namen eines Produkt-REST-Endpunkts aus, z. B. /dogs - GET.
4. Wählen Sie Seite bearbeiten aus.
5. Wählen Sie Funktionen testen aus, damit Kunden Ihre API in Ihrem Portal aufrufen können.
6. Wählen Sie Änderungen speichern aus.
7. Sie müssen alle Portale, die dieses Portalprodukt verwenden, erneut veröffentlichen, damit die Änderungen wirksam werden.

Löschen Sie ein Portalprodukt in API Gateway

Wenn Sie ein Portalprodukt löschen, kann es nicht wiederhergestellt werden.

Löschen Sie ein Portalprodukt

Das folgende Verfahren zeigt, wie Sie ein Portalprodukt löschen.

Um ein Portalprodukt zu löschen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie ein Portalprodukt aus.
4. Wählen Sie Löschen aus.
5. Bestätigen Sie Ihre Wahl und wählen Sie Löschen aus.
6. Sie müssen alle Portale, die dieses Portalprodukt verwenden, erneut veröffentlichen, damit die Änderungen wirksam werden.

Erstellen Sie ein Portal in API Gateway

Ein Portal ist eine Sammlung von Produkten. Sie kontrollieren die Autorisierung, das Branding und die Veröffentlichung Ihres Portals. Wenn Sie ein Portal erstellen, lautet der Veröffentlichungsstatus `Disabled`. Das bedeutet, dass Ihr Portal im Internet nicht auffindbar ist. Damit Verbraucher auf Ihr Portal zugreifen können, müssen Sie es veröffentlichen. Bevor Sie Ihr Portal veröffentlichen, können Sie es auch in der Vorschau anzeigen.

API Gateway unterstützt Portale in den folgenden Browsern:

- Firefox
- Google Chrome
- Microsoft Edge
- Safari

Wählen Sie einen Domainnamen

Standardmäßig wird Ihre Portal-URL unter dem folgenden Domainnamen gehostet, der Amazon gehört:

```
https://p-portalId.apigw-portal.us-east-1.on.aws
```

Um den Domainnamen anzupassen, können Sie einen Domainnamen, den Sie besitzen, und ein SSL-Zertifikat für die benutzerdefinierte Domain angeben. Das Zertifikat muss mit [AWS Certificate Manager](#) (ACM) in USA Ost (Nord-Virginia) verwaltet und von Amazon CloudFront unterstützt werden. Weitere Informationen finden Sie unter [Anforderungen für die Verwendung von SSL/TLS Zertifikaten mit CloudFront](#).

Wenn Sie einen benutzerdefinierten Domainnamen verwenden, erstellt API Gateway eine CloudFront Distribution, die während der Übertragung mit Ihrem ACM-Zertifikat gesichert ist. Anschließend stellt API Gateway einen Aliaseintrag bereit, den Sie zu Ihrer DNS-Konfiguration hinzufügen. Der Aliaseintrag ist der Domainname, der Amazon gehört. Sie müssen diesen Aliaseintrag hinzufügen, um Traffic an die CloudFront Distribution für Ihre benutzerdefinierte Domain weiterzuleiten. Der von Ihnen angegebene Domainname muss eindeutig sein und darf nicht bereits in der Aliasliste einer bestehenden CloudFront Distribution verwendet werden.

Wenn Sie Ihren eigenen Domainnamen verwenden, verwendet API Gateway die TLS_1_2 Sicherheitsrichtlinie, um Ihr Portal zu schützen. Weitere Informationen finden Sie unter [Unterstützte Protokolle und Chiffren zwischen Zuschauern](#). Sie können diesen Wert nicht ändern.

Erstellen Sie ein Portal

Sie verwenden die API Gateway Gateway-Konsole, um Ihr Portal zu erstellen und zu konfigurieren. Die folgenden Verfahren führen Sie durch alle erforderlichen Schritte. Sie können am Ende des ersten Verfahrens auch „Zur Überprüfung überspringen“ und „Erstellen“ wählen, um ein leeres Portal zu erstellen.

Das folgende Verfahren zeigt, wie Sie ein Portal erstellen. Zunächst definieren Sie Ihre Portaldetails.

Um Ihre Portaldetails zu definieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.

3. Wählen Sie **Create Portal (Portal erstellen)** aus.
4. Geben Sie unter **Portaldetails** für **Portalname** einen Namen ein.
5. Geben Sie unter **Portalbeschreibung** eine Beschreibung ein.
6. Wählen Sie für die **Portalautorisierung** aus, wie Sie Ihr Portal autorisieren möchten.
 - Wenn Sie zum ersten Mal ein Portal erstellen, empfehlen wir Ihnen, **None** zu wählen. Wenn Sie Ihrem Portal eine Produktion APIs hinzufügen möchten, verwenden Sie die Option **Autorisierung erforderlich**.
 - Wenn Sie festlegen möchten, dass sich Portalnutzer für die Nutzung Ihres Portals authentifizieren müssen, wählen Sie **Autorisierung erforderlich** aus und gehen Sie wie folgt vor:
 - a. Wählen Sie für den **Amazon Cognito Cognito-Benutzerpool** Ihren Benutzerpool aus.

Der Benutzerpool muss sich in derselben Region wie Ihr Portal befinden.
 - b. Wählen Sie für **Anwendungsclient-ID** die **App-Client-ID** aus.

In Ihrem App-Client müssen Sie die **Callback-URL** auf die **Standard-URL** des Portals festlegen.
 - c. Geben Sie für **Benutzerpool-Domäne** die **Benutzerpool-Domäne** ein. Weitere Informationen finden Sie unter [Konfiguration einer Benutzerpool-Domäne](#).
7. Wählen Sie für die **Domänenkonfiguration** den **Domainnamen** für Ihr Portal aus.
 - Wenn Sie zum ersten Mal ein Portal erstellen, empfehlen wir Ihnen, **Standarddomäne** zu wählen. Sobald Sie Ihr Portal veröffentlicht haben, ist es unter `verfügbarhttps://p-portalId.apigw-portal.us-east-1.on.aws`.
 - Um einen benutzerdefinierten Domainnamen anzugeben, den Sie besitzen, wählen Sie **Benutzerdefinierter Domainname** aus und gehen Sie wie folgt vor:
 - a. Geben Sie unter **Domainname** einen vollqualifizierten Domainnamen ein.
 - b. Wählen Sie für ein **ACM-Zertifikat** ein **ACM-Zertifikat** aus, das den Domainnamen abdeckt.

Sie müssen den **Aliaseintrag**, den API Gateway bereitstellt, zu Ihrem **DNS-Eintrag** hinzufügen, um Traffic von Ihrem Portal aus bereitzustellen.
8. Um Ihrem Portal einen **CloudWatch RUM-App-Monitor** hinzuzufügen, wählen Sie für **RUM-App-Monitor** einen **App-Monitor** aus. Weitere Informationen finden Sie unter [CloudWatch RUM](#).
9. Wählen Sie **Weiter** aus.

Nachdem Sie Ihre Portaldetails definiert haben, fügen Sie Portalprodukte hinzu. Sie können die Portalprodukte in Ihrem Portal jederzeit ändern, müssen Ihr Portal jedoch erneut veröffentlichen, damit die Änderungen wirksam werden.

Um Ihre Portalprodukte auszuwählen

1. Wählen Sie aus der Ressourcenliste die Produkte aus, die Sie Ihrem Portal hinzufügen möchten.
2. Wählen Sie Weiter aus.

Nachdem Sie Ihre Produkte ausgewählt haben, passen Sie das Portaldesign an. Um ein Portal-Logo hochzuladen, müssen Sie ein Bild auf Amazon S3 hochgeladen haben. Sie können diese Werte jederzeit ändern, müssen Ihr Portal jedoch erneut veröffentlichen, damit die Änderungen wirksam werden.

(Optional) Um Ihr Portal anzupassen

1. Geben Sie für das Portal-Logo den S3-URI für Ihre Datei an.

Nur .svg, .png, oder .jpg werden unterstützt.

2. Um das Design Ihres Portals anzupassen, haben Sie zwei Möglichkeiten:
 - a. Sie können das Portaldesign auswählen, um ein vorgefertigtes Farbdesign auf Ihr Portal anzuwenden.
 - b. Sie können die Farbauswahlmöglichkeiten verwenden, um alle Ihre Portal-Designoptionen auszuwählen.
3. Wählen Sie Weiter aus.

Nachdem Sie das Portaldesign ausgewählt haben, überprüfen Sie die Informationen für das Portal.

Um Ihr Portal zu überprüfen

- Überprüfen Sie Ihre Einstellungen auf der Seite „Überprüfen und erstellen“. Wenn Sie mit den Einstellungen zufrieden sind, klicken Sie auf Create (Erstellen).

Wenn Sie einen Amazon Cognito Cognito-Benutzerpool verwenden, um den Zugriff auf Ihr Portal zu steuern, müssen Sie die Callback-URL im App-Client auf die Standard-URL des Portals festlegen. Weitere Informationen finden Sie unter [Anwendungsspezifische Einstellungen](#) mit App-Clients.

Ihr Portal steht Verbrauchern nicht mehr zur Verfügung, nachdem Sie es erstellt haben. Sie müssen Ihr Portal veröffentlichen, damit es im Internet zugänglich ist. Weitere Informationen finden Sie unter [the section called “Veröffentlichen Sie ein Portal”](#).

Aktualisieren Sie ein Portal in API Gateway

Sie können das Thema, das Logo oder die Portalprodukte in Ihrem Portal aktualisieren. Wenn Sie Ihr Portal aktualisieren, werden die Änderungen erst wirksam, wenn Sie Ihr Portal erneut veröffentlichen.

Aktualisieren Sie ein Portal

Das folgende Verfahren zeigt, wie Sie das Design eines Portals aktualisieren.

Um ein Portaldesign zu aktualisieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wählen Sie auf der Registerkarte Portaldesign die Option Bearbeiten aus.
5. Um das Portal-Logo zu aktualisieren, wählen Sie Portal-Logo aktualisieren. Sie können entweder das vorhandene Logo entfernen oder ein neues Logo verwenden.
6. Nehmen Sie beliebige Themen- oder Hintergrundaktualisierungen an Ihrem Portaldesign vor.
7. Wählen Sie Änderungen speichern aus.
8. (Optional) Wählen Sie Portalvorschau generieren, um Ihre Änderungen zu sehen.

Nachdem Sie Ihre Änderungen bestätigt haben, veröffentlichen Sie Ihr Portal.

9. Wählen Sie Portal veröffentlichen aus.
10. Geben Sie eine Beschreibung für Ihr Portal-Update ein und wählen Sie Veröffentlichen.

Vorschau eines Portals in API Gateway

Sie können eine Vorschau Ihres Portals anzeigen, bevor Sie es veröffentlichen. API Gateway erstellt eine kurzlebige URL, mit der Sie eine Vorschau der Version Ihres Portals anzeigen können, die Kunden nach der Veröffentlichung Ihres Portals sehen werden.

Überlegungen

Die folgenden Überlegungen können sich darauf auswirken, wie Sie ein Portal in der Vorschau anzeigen:

- Try it ist für Ihr Vorschauportal nicht aktiviert. Weitere Informationen finden Sie unter [the section called “Aktivieren Sie Try it für einen Produkt-REST-Endpunkt in Ihrem Portal”](#).

Eine Vorschau eines Portals anzeigen

Das folgende Verfahren zeigt, wie Sie eine Vorschau eines Portals anzeigen können.

Um eine Vorschau eines Portals anzuzeigen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wählen Sie Portalvorschau generieren aus.

Aktualisieren Sie Ihre Seite nicht.

5. Nachdem API Gateway Ihre Vorschau generiert hat, wählen Sie Portalvorschau öffnen.

Ihre Portalvorschau wird in einem neuen Tab geöffnet.

Veröffentlichen Sie ein Portal in API Gateway

Damit API-Nutzer auf Ihr Portal zugreifen können, müssen Sie es veröffentlichen. Eine Portal-URL kann von jedem im Internet entdeckt werden. Wir empfehlen, dass Sie eine Vorschau Ihres Portals anzeigen und es sichern, bevor Sie es veröffentlichen.

Überlegungen

Es kann einige Minuten dauern, bis API Gateway Ihr Portal veröffentlicht. Sie können den Veröffentlichungsstatus in der Konsole überwachen.

Veröffentlichen Sie ein Portal

Das folgende Verfahren zeigt, wie Sie ein Portal veröffentlichen.

Um ein Portal zu veröffentlichen

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wählen Sie Portal veröffentlichen.
5. (Optional) Geben Sie unter Beschreibung der Änderungen eine Beschreibung Ihrer Änderung ein.

Wenn Sie ein Portal veröffentlichen, empfehlen wir, dass Sie immer eine kurze Beschreibung Ihrer Änderungen angeben.

6. Wählen Sie Publish.

Es dauert einige Minuten, bis API Gateway die Veröffentlichung Ihres Portals abgeschlossen hat. API Gateway stellt einen Link zu Ihrem Portal bereit, sobald dieser verfügbar ist.

Um Ihr Portal zu löschen, müssen Sie es zuerst deaktivieren. Weitere Informationen finden Sie unter [the section called “Deaktivieren Sie ein Portal”](#).

Verwenden Sie ein API-Gateway-Portal

Nachdem Sie Ihr Portal veröffentlicht haben, ist es im Internet verfügbar. Um Ihr Portal zu verwenden, kopieren Sie den Standard-Domainnamen oder den Domainnamen des Portals und geben Sie ihn in einen Browser ein. Wenn Ihr Portal einen Domainnamen verwendet, kann der Standard-Domainname nicht für den Zugriff auf Ihr Portal verwendet werden.

Verwenden Sie ein Portal

Das folgende Verfahren zeigt, wie Sie auf ein Portal zugreifen und try it für einen Produkt-REST-Endpunkt verwenden.

Um ein Portal zu verwenden

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Kopieren Sie den Standard-Domainnamen oder den Domainnamen des Portals und geben Sie ihn in einen Browser ein.
5. Auf der Startseite des Portals können Sie nach Portalprodukten suchen und Produktseiten entdecken. Verwenden Sie die Suche oben im Portal, um den gesamten Inhalt Ihres Portals zu durchsuchen. Verwenden Sie die Suchleiste unter Produkte, um nach Portalprodukten zu suchen.
6. Wählen Sie ein Portalprodukt aus.
7. Wählen Sie einen Produkt-REST-Endpunkt aus.
8. Geben Sie im Fenster Testen die Parameter ein, um zu versuchen, Ihre API aufzurufen.
9. Wählen Sie Probieren Sie es aus. API Gateway ruft Ihre API auf und zeigt die Ergebnisse im Fenster an.

Beheben Sie Fehler in Ihrem Portal

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung eines Portals auftreten können.

Die Website der Portal-Webseite kann nicht erreicht werden

Wenn Sie eine Fehlermeldung erhalten, dass die Portal-Webseitenwebsite nicht erreicht werden kann, stellen Sie sicher, dass Sie Ihr Portal veröffentlicht haben. Wenn Sie einen Domainnamen verwendet haben, stellen Sie sicher, dass Sie den Aliaseintrag hinzugefügt haben, den API Gateway bereitgestellt hat, um den Traffic an die CloudFront Distribution für Ihre benutzerdefinierte Domain weiterzuleiten.

Versuch, es ist fehlgeschlagen

Wenn Sie Try it nicht für einen Produkt-REST-Endpunkt verwenden können, stellen Sie sicher, dass der Test-Status lautet ACTIVE. Stellen Sie außerdem sicher, dass Sie die richtigen Parameter eingegeben haben, um Ihre API aufzurufen.

Seiten werden in Ihrem Portal nicht angezeigt

Wenn Sie eine REST-Produktendpunktseite oder eine Produktseite erstellt haben und diese in Ihrem Portal nicht sichtbar ist, stellen Sie sicher, dass es sich bei Ihren Seiten nicht um Entwürfe handelt. Gehen Sie wie folgt vor, um dies zu überprüfen:

Um zu überprüfen, ob es sich bei Ihren Seiten nicht um Entwürfe handelt

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portal-Produkte aus.
3. Wählen Sie Ihr Portalprodukt aus.
4. Vergewissern Sie sich auf der Registerkarte Dokumentation, dass sich Ihre Seiten nicht im Abschnitt Dokumentationsentwürfe oder API-Referenzseiten für Entwürfe, sondern in den Seiten für benutzerdefinierte Dokumentation oder API-Referenzseiten befinden.

Deaktivieren Sie ein Portal in API Gateway

Wenn Sie ein Portal deaktivieren, ist Ihr Portal nicht mehr im Internet verfügbar. Nachdem Sie ein Portal deaktiviert haben, können Sie das Portal entweder löschen oder bei Bedarf erneut veröffentlichen.

Deaktivieren Sie ein Portal

Das folgende Verfahren zeigt, wie ein Portal deaktiviert wird.

Um ein Portal zu deaktivieren

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wählen Sie Aktionen und dann Portal deaktivieren aus.
5. Bestätigen Sie Ihre Auswahl und wählen Sie Portal deaktivieren.

Der Status „Veröffentlichen“ wird auf Deaktiviert geändert.

Um den Deaktivierungsstatus des Portals zu entfernen, veröffentlichen Sie das Portal erneut.

Löschen Sie ein Portal in API Gateway

Wenn Sie ein Portal löschen, kann es nicht wiederhergestellt werden und Sie müssen es zuerst deaktivieren. Wenn Sie Ihr Portal aus dem Internet entfernen möchten, können Sie es deaktivieren. Auf diese Weise können Sie Ihr Portal ändern und es später erneut veröffentlichen.

Löschen Sie ein Portal

Das folgende Verfahren zeigt, wie Sie ein Portal löschen.

So löschen Sie ein Portal

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie im Hauptnavigationsbereich Portals aus.
3. Wählen Sie ein Portal aus.
4. Wenn Ihr Portal nicht deaktiviert ist, deaktivieren Sie es, indem Sie Aktionen, Portal deaktivieren wählen.

Es dauert einige Minuten, bis Ihr Portal deaktiviert ist. Sie können den Veröffentlichungsstatus überwachen, um zu sehen, wann der Veröffentlichungsstatus Ihres Portals deaktiviert ist.

5. Nachdem Sie die Veröffentlichung Ihres Portals rückgängig gemacht haben, können Sie es löschen. Um Ihr Portal zu löschen, wählen Sie Aktionen, Portal löschen. Bestätigen Sie Ihre Wahl und wählen Sie Löschen aus.

API-Gateway-HTTP-APIs

REST-APIs und HTTP-APIs sind beide RESTful-API-Produkte. REST-APIs unterstützen mehr Funktionen als HTTP-APIs. HTTP-APIs sind mit minimalen Funktionen ausgestattet, sodass sie zu einem niedrigeren Preis angeboten werden können. Weitere Informationen finden Sie unter [the section called “Wählen Sie zwischen REST APIs und HTTP APIs”](#).

Sie können HTTP-APIs verwenden, um Anfragen an AWS Lambda-Funktionen oder an einen routingfähigen HTTP-Endpunkt zu senden. Beispielsweise können Sie eine HTTP-API erstellen, die in eine Lambda-Funktion auf dem Backend integriert wird. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion und gibt die Antwort der Funktion an den Client zurück.

HTTP-APIs unterstützen die [OpenID Connect](#)- und [OAuth 2.0](#)-Autorisierung. Sie verfügen über integrierten Support für Cross-Origin Resource Sharing (CORS) und automatische Bereitstellungen.

Sie können HTTP-APIs mithilfe der AWS-Managementkonsole, der AWS CLI, APIs, CloudFormation oder SDKs erstellen.

Themen

- [Entwickeln Sie HTTP APIs in API Gateway](#)
- [HTTP-APIs für Kundenaufrufe veröffentlichen](#)
- [Schutz Ihrer HTTP-APIs in API Gateway](#)
- [Überwachen Sie HTTP APIs im API Gateway](#)
- [Fehlerbehebung bei Problemen mit HTTP-APIs in API Gateway](#)

Entwickeln Sie HTTP APIs in API Gateway

Dieser Abschnitt enthält Einzelheiten zu den API Gateway Gateway-Funktionen, die Sie bei der Entwicklung Ihres API Gateway benötigen APIs.

Während Sie Ihre API Gateway-API entwickeln, entscheiden Sie sich für eine Reihe von Merkmalen Ihrer API. Diese Eigenschaften hängen davon ab, wofür Ihre API verwendet werden soll. So könnte es beispielsweise sein, dass Sie es nur bestimmten Clients gestatten möchten, die API aufzurufen. Vielleicht soll die API aber auch für alle verfügbar sein. Vielleicht benötigen Sie einen API-Aufruf, um eine Lambda-Funktion auszuführen, eine Datenbankabfrage durchzuführen oder eine Anwendung aufzurufen.

Themen

- [Erstellen einer HTTP-API](#)
- [Routen für HTTP APIs in API Gateway erstellen](#)
- [IP-Adresstypen für HTTP-APIs in API Gateway](#)
- [Steuern und verwalten Sie den Zugriff auf HTTP APIs in API Gateway](#)
- [Erstellen von Integrationen für HTTP-APIs in API Gateway](#)
- [CORS für HTTP APIs im API Gateway konfigurieren](#)
- [API-Anforderungen und -Antworten für HTTP-APIs in API Gateway transformieren](#)
- [Verwenden Sie OpenAPI-Definitionen für HTTP APIs in API Gateway](#)

Erstellen einer HTTP-API

Um eine funktionale API erstellen zu können, benötigen Sie mindestens eine Route, eine Integration, eine Phase und eine Bereitstellung.

Die folgenden Beispiele zeigen, wie Sie eine API mit einer AWS Lambda oder HTTP-Integration, einer Route und einer Standardstufe erstellen, die so konfiguriert ist, dass Änderungen automatisch bereitgestellt werden.

In diesem Leitfaden wird davon ausgegangen, dass Sie bereits mit API Gateway und Lambda vertraut sind. Einen ausführlicheren Leitfaden finden Sie unter [Erste Schritte](#).

Themen

- [Erstellen Sie eine HTTP-API mithilfe der AWS-Managementkonsole](#)
- [Erstellen Sie eine HTTP-API mithilfe der AWS CLI](#)

Erstellen Sie eine HTTP-API mithilfe der AWS-Managementkonsole

1. Öffnen Sie die [API Gateway-Konsole](#).
2. Wählen Sie Create API (API erstellen) aus.
3. Wählen Sie unter HTTP-API die Option Entwickeln aus.
4. Wählen Sie Add integration (Integration hinzufügen) und wählen Sie dann eine AWS Lambda - Funktion oder geben Sie einen HTTP-Endpunkt ein.
5. Geben Sie im Feld Name einen Namen für Ihre API ein.

6. Wählen Sie Review and create.
7. Wählen Sie Create aus.

Jetzt kann Ihre API aufgerufen werden. Sie können Ihre API testen, indem Sie die Aufruf-URL in einem Browser eingeben oder cURL verwenden.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Erstellen Sie eine HTTP-API mithilfe der AWS CLI

Sie können Quick Create verwenden, um eine API mit einer Lambda- oder HTTP-Integration, eine Catch-All-Standardroute und eine Standardphase zu erstellen, die für die automatische Bereitstellung von Änderungen konfiguriert ist. Der folgende [create-api](#)-Befehl verwendet Quick Create, um eine API zu erstellen, die in eine Lambda-Funktion im Backend integriert werden kann.

Note

Um eine Lambda-Integration aufzurufen, muss API Gateway über die erforderlichen Berechtigungen verfügen. Sie können eine ressourcenbasierte Richtlinie oder eine IAM-Rolle verwenden, um API-Gateway Berechtigungen zum Aufrufen einer Lambda-Funktion zu erteilen. Weitere Informationen finden Sie unter [AWS Lambda Berechtigungen](#) im AWS Lambda Entwicklerhandbuch.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target  
arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Jetzt kann Ihre API aufgerufen werden. Sie können Ihre API testen, indem Sie die Aufruf-URL in einem Browser eingeben oder cURL verwenden.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Routen für HTTP APIs in API Gateway erstellen

Routen leiten eingehende API-Anfragen an Backend-Ressourcen. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispiel, GET /pets. Sie können

spezifische HTTP-Methoden für Ihre Route definieren. Sie können auch die Methode ANY verwenden, um alle Methoden abzugleichen, die Sie nicht für eine Ressource definiert haben. Sie können eine `$default`-Route erstellen, die als Catch-All für Anfragen fungiert, die nicht mit anderen Routen übereinstimmen.

Note

API Gateway dekodiert URL-kodierte Anforderungsparameter vor deren Übergabe an Ihre Backend-Integration.

Arbeiten mit Pfadvariablen

Sie können Pfadvariablen in HTTP-API-Routen verwenden.

Beispielsweise fängt die Route `GET /pets/{petID}` eine GET-Anforderung ab, die ein Client an `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6` sendet.

Eine gierige Pfadvariable fängt alle untergeordneten Ressourcen einer Route ab. Um eine gierige Pfadvariable zu erstellen, fügen Sie `+` dem Variablennamen hinzu, z. B. `{proxy+}`. Die gierige Pfadvariable muss am Ende des Ressourcenpfads stehen.

Arbeiten mit Abfragezeichenfolgenparametern

API Gateway sendet standardmäßig Abfragezeichenfolgenparameter an Ihre Backend-Integration, sofern sie in einer Anforderung an eine HTTP-API enthalten sind.

Wenn beispielsweise ein Client eine Anforderung an `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog` sendet, werden die Abfragezeichenfolgenparameter `?id=4&type=dog` an Ihre Integration gesendet.

Arbeiten mit der `$default`-Route

Die `$default`-Route fängt Anforderungen ab, die nicht explizit mit anderen Routen in Ihrer API übereinstimmen.

Wenn die `$default`-Route eine Anforderung empfängt, sendet API Gateway den vollständigen Anforderungspfad an die Integration. Beispielsweise können Sie eine API nur mit einer `$default`-Route erstellen und sie in der Methode ANY mit dem HTTP-Endpunkt `https://petstore-demo-endpoint.execute-api.com` integrieren. Wenn Sie eine Anforderung an `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout` senden, sendet API

Gateway eine Anforderung an `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Weitere Informationen zu HTTP-Integrationen finden Sie unter [Erstellen Sie HTTP-Proxyintegrationen für HTTP APIs](#).

Weiterleiten von API-Anforderungen

Wenn ein Client eine API-Anforderung sendet, bestimmt API Gateway zuerst, an welche [Stufe](#) die Anforderung weitergeleitet werden soll. Wenn die Anforderung explizit mit einer Stufe übereinstimmt, sendet API Gateway die Anforderung an diese Stufe. Wenn keine Stufe vollständig mit der Anforderung übereinstimmt, sendet API Gateway die Anforderung an die `$default`-Stufe. Wenn es keine `$default` Phase gibt, kehrt die API zurück `{"message": "Not Found"}` und generiert keine CloudWatch Protokolle.

Nachdem Sie eine Stufe ausgewählt haben, wählt API Gateway eine Route aus. API Gateway wählt die Route mit der spezifischsten Übereinstimmung aus, wobei die folgenden Prioritäten verwendet werden:

1. Vollständige Übereinstimmung für eine Route und Methode.
2. Übereinstimmung für eine Route und eine Methode mit einer gierigen Pfadvariable (`{proxy+}`).
3. Die `$default`-Route.

Wenn keine Routen mit einer Anforderung übereinstimmen, gibt API Gateway `{"message": "Not Found"}` an den Client zurück.

Betrachten Sie beispielsweise eine API mit einer `$default`-Stufe und den folgenden Beispielrouten:

1. GET `/pets/dog/1`
2. GET `/pets/dog/{id}`
3. GET `/pets/{proxy+}`
4. ANY `/ {proxy+}`
5. `$default`

In der folgenden Tabelle wird zusammengefasst, wie API Gateway Anforderungen an die Beispielrouten weiterleitet.

Anfrage	Ausgewählte Route	Erklärung
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/pets/dog/1	GET /pets/dog/1	Die Anforderung entspricht vollständig dieser statischen Route.
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/pets/dog/2	GET /pets/dog/{id}	Die Anforderung stimmt vollständig mit dieser Route überein.
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/pets/cat/1	GET /pets/{proxy+}	Die Anforderung stimmt nicht vollständig mit einer Route überein. Die Route mit einer GET-Methode und einer gierigen Pfadvariablen fängt diese Anforderung ab.
POST https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/test/5	ANY /{proxy+}	Die Methode ANY entspricht allen Methoden, die Sie nicht für eine Route definiert haben. Routen mit gierigen Pfadvariablen haben eine höhere Priorität als die \$default-Route.

IP-Adresstypen für HTTP-APIs in API Gateway

Beim Erstellen einer API geben Sie den Typ der IP-Adressen an, die Ihre API aufrufen dürfen. Sie können IPv4 auswählen, um IPv4-Adressen für den Aufruf Ihrer API zuzulassen, oder Sie können Dualstack auswählen, um sowohl IPv4- als auch IPv6-Adressen zuzulassen. Es kann sinnvoll sein, den IP-Adresstyp auf Dualstack zu setzen, um Engpässe im IP-Adressraum zu vermeiden oder Ihre Sicherheitslage zu verbessern. Weitere Informationen zu den Vorteilen eines Dualstack-IP-Adresstyps finden Sie unter [IPv6 in AWS](#).

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können Ihre Verwendung von IP-Adresstypen beeinflussen:

- Der Standard-IP-Adresstyp für HTTP-APIs ist IPv4.
- Wenn Sie den IP-Adresstyp einer bestehenden API von IPv4 in Dualstack ändern, stellen Sie sicher, dass alle Richtlinien, die den Zugriff auf Ihre APIs steuern, aktualisiert wurden, um IPv6-Aufrufe zu berücksichtigen. Wenn Sie den IP-Adresstyp ändern, wird die Änderung sofort wirksam.
- Ihre API kann einem benutzerdefinierten Domainnamen mit einem anderen IP-Adresstyp als Ihre API zugewiesen werden. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.

Ändern des IP-Adresstyps einer HTTP-API

Sie können den IP-Adresstyp ändern, indem Sie die API-Konfiguration aktualisieren. Dies ist über die AWS-Managementkonsole, die AWS CLI, CloudFormation oder ein AWS SDK möglich. Wenn Sie den IP-Adresstyp der API ändern, müssen Sie Ihre API nicht erneut bereitstellen, damit die Änderungen wirksam werden.

AWS-Managementkonsole

So ändern Sie den IP-Adresstyp einer HTTP-API

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie unter API-Einstellungen die Option Bearbeiten aus.
4. Wählen Sie unter „IP-Adresstyp“ entweder IPv4 oder Dualstack aus.
5. Wählen Sie Speichern aus.

Die Änderung der API-Konfiguration wird sofort wirksam.

AWS CLI

Der folgende [update-api](#)-Befehl aktualisiert eine API, sodass sie den IP-Adresstyp Dualstack verwendet:

```
aws apigatewayv2 update-api \
```

```
--api-id abcd1234 \  
--ip-address-type dualstack
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",  
  "ApiId": "abcd1234",  
  "ApiKeySelectionExpression": "$request.header.x-api-key",  
  "CreateDate": "2025-02-04T22:20:20+00:00",  
  "DisableExecuteApiEndpoint": false,  
  "Name": "My-HTTP-API",  
  "ProtocolType": "HTTP",  
  "RouteSelectionExpression": "$request.method $request.path",  
  "Tags": {},  
  "NotificationUri": [],  
  "IpAddressType": "dualstack"  
}
```

Steuern und verwalten Sie den Zugriff auf HTTP APIs in API Gateway

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre HTTP-API:

- Lambda-Autorisierer verwenden Lambda-Funktionen, um den Zugriff auf zu kontrollieren. APIs Weitere Informationen finden Sie unter [Steuern Sie den Zugriff auf HTTP APIs mit AWS Lambda Autorisierern](#).
- JWT-Autorisierer verwenden JSON-Webtoken, um den Zugriff auf zu kontrollieren. APIs Weitere Informationen finden Sie unter [Steuern Sie den Zugriff auf HTTP APIs mit JWT-Autorisierern in API Gateway](#).
- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen. Mithilfe von IAM-Rollen und -Richtlinien können Sie steuern, wer Ihre APIs Rollen erstellen und verwalten darf und wer sie aufrufen kann. Weitere Informationen finden Sie unter [Steuern Sie den Zugriff auf HTTP APIs mit der IAM-Autorisierung in API Gateway](#).

Um Ihre Sicherheitslage zu verbessern, empfehlen wir, für alle Routen Ihrer HTTP-API einen Genehmiger zu konfigurieren. Dies kann erforderlich sein, um verschiedene Compliance-

Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

Steuern Sie den Zugriff auf HTTP APIs mit AWS Lambda Autorisierern

Sie verwenden einen Lambda-Genehmiger, um den Zugriff auf Ihre HTTP-API mit einer Lambda-Funktion zu steuern. Wenn dann ein Client Ihre API aufruft, ruft API Gateway Ihre Lambda-Funktion auf. API Gateway verwendet die Antwort Ihrer Lambda-Funktion, um festzustellen, ob der Client auf Ihre API zugreifen kann.

Nutzlastformatversion

Die Nutzlastformatversion des Genehmigers gibt das Format der Daten an, die API Gateway an einen Lambda-Genehmiger sendet, und wie API Gateway die Antwort von Lambda interpretiert. Wenn Sie keine Version im Payload-Format angeben, wird standardmäßig die neueste Version AWS-Managementkonsole verwendet. Wenn Sie einen Lambda-Autorisierer mithilfe des AWS CLI, CloudFormation, oder eines SDK erstellen, müssen Sie einen angeben. `authorizerPayloadFormatVersion` Die unterstützten Werte sind `1.0` und `2.0`.

Wenn Sie Kompatibilität mit REST benötigen APIs, verwenden Sie Version. `1.0`

Die folgenden Beispiele zeigen die Struktur jeder Nutzlastformatversion.

2.0

```
{
  "version": "2.0",
  "type": "REQUEST",
  "routeArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": ["user1", "123"],
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": ["cookie1", "cookie2"],
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
```

```

},
"requestContext": {
  "accountId": "123456789012",
  "apiId": "api-id",
  "authentication": {
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  }
},
"domainName": "id.execute-api.us-east-1.amazonaws.com",
"domainPrefix": "id",
"http": {
  "method": "POST",
  "path": "/my/path",
  "protocol": "HTTP/1.1",
  "sourceIp": "IP",
  "userAgent": "agent"
},
"requestId": "id",
"routeKey": "$default",
"stage": "$default",
"time": "12/Mar/2020:19:03:58 +0000",
"timeEpoch": 1583348638390
},
"pathParameters": { "parameter1": "value1" },
"stageVariables": { "stageVariable1": "value1", "stageVariable2": "value2" }
}

```

1.0

```

{
  "version": "1.0",
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
  "identitySource": "user1,123",

```



```
"authorizationToken": "user1,123",
"resource": "/request",
"path": "/request",
"httpMethod": "GET",
"headers": {
  "X-AMZ-Date": "20170718T062915Z",
  "Accept": "*/*",
  "HeaderAuth1": "headerValue1",
  "CloudFront-Viewer-Country": "US",
  "CloudFront-Forwarded-Proto": "https",
  "CloudFront-Is-Tablet-Viewer": "false",
  "CloudFront-Is-Mobile-Viewer": "false",
  "User-Agent": "...",
},
"queryStringParameters": {
  "QueryString1": "queryValue1"
},
"pathParameters": {},
"stageVariables": {
  "StageVar1": "stageValue1"
},
"requestContext": {
  "path": "/request",
  "accountId": "123456789012",
  "resourceId": "05c7jb",
  "stage": "test",
  "requestId": "...",
  "identity": {
    "apiKey": "...",
    "sourceIp": "...",
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  }
},
"resourcePath": "/request",
"httpMethod": "GET",
"apiId": "abcdef123"
```

```
}  
}
```

Antwortformat des Lambda-Genehmigers

Die Nutzlastformatversion bestimmt auch die Struktur der Antwort, die von Ihrer Lambda-Funktion zurückgegeben werden muss.

Lambda-Funktionsantwort für Format 1.0

Wenn Sie Formatversion 1.0 auswählen, müssen Lambda-Genehmiger eine IAM-Richtlinie zurückgeben, die den Zugriff auf Ihre API-Route zulässt oder verweigert. Sie können die IAM-Standardrichtliniensyntax in der Richtlinie verwenden. Beispiele für IAM-Richtlinien finden Sie unter [the section called “Kontrollieren des Zugriffs für den API-Aufruf”](#). Mit können Sie Kontexteigenschaften an Lambda-Integrationen oder -Zugriffsprotokolle übergeben `$context.authorizer.property`. Das context-Objekt ist optional und `claims` ist ein reservierter Platzhalter, der nicht als Kontextobjekt verwendet werden kann. Weitere Informationen hierzu finden Sie unter [the section called “Protokollierungsvariablen”](#).

Example

JSON

```
{  
  "principalId": "abcdef",  
  "policyDocument": {  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Action": "execute-api:Invoke",  
        "Effect": "Allow|Deny",  
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-resources}"]  
      }  
    ],  
  },  
  "context": {  
    "exampleKey": "exampleValue"  
  }  
}
```

Lambda-Funktionsantwort für Format 2.0

Wenn Sie Formatversion 2.0 auswählen, können Sie einen booleschen Wert oder eine IAM-Richtlinie zurückgeben, die die IAM-Standardrichtliniensyntax Ihrer Lambda-Funktion verwendet. Um einen booleschen Wert zurückzugeben, aktivieren Sie einfache Antworten für den Genehmiger. Die folgenden Beispiele veranschaulichen das Format, das Sie für die Rückgabe Ihrer Lambda-Funktion codieren müssen. Das Objekt `context` ist optional. Mit können Sie Kontexteigenschaften an Lambda-Integrationen oder -Zugriffsprotokolle übergeben `$context.authorizer.property`. Weitere Informationen hierzu finden Sie unter [the section called "Protokollierungsvariablen"](#).

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-resources}"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Beispiel für Lambda-Genehmiger-Funktionen

Die Lambda-Funktionen des folgenden Node.js-Beispiels veranschaulichen die erforderlichen Antwortformate, die Sie von Ihrer Lambda-Funktion für Formatversion 2.0 der Nutzlast zurückgeben müssen.

Simple response - Node.js

```
export const handler = async(event) => {
  let response = {
    "isAuthorized": false,
    "context": {
      "stringKey": "value",
      "numberKey": 1,
      "booleanKey": true,
      "arrayKey": ["value1", "value2"],
      "mapKey": {"value1": "value2"}
    }
  };

  if (event.headers.authorization === "secretToken") {
    console.log("allowed");
    response = {
      "isAuthorized": true,
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": {"value1": "value2"}
      }
    };
  }

  return response;
};
```

Simple response - Python

```
import json
```

```
def lambda_handler(event, context):
    response = {
        "isAuthorized": False,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                "isAuthorized": True,
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

IAM policy - Node.js

```
export const handler = async(event) => {
    if (event.headers.authorization == "secretToken") {
        console.log("allowed");
        return {
            "principalId": "abcdef", // The principal user identification associated with
            the token sent by the client.
            "policyDocument": {
                "Version": "2012-10-17",
```

```

        "Statement": [{
            "Action": "execute-api:Invoke",
            "Effect": "Allow",
            "Resource": event.routeArn
        }]
    },
    "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
    }
};
}
else {
    console.log("denied");
    return {
        "principalId": "abcdef", // The principal user identification associated with
the token sent by the client.
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event.routeArn
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": true,
            "arrayKey": ["value1", "value2"],
            "mapKey": { "value1": "value2" }
        }
    };
}
};

```

IAM policy - Python

```
import json
```

```
def lambda_handler(event, context):
    response = {
        # The principal user identification associated with the token sent by
        # the client.
        "principalId": "abcdef",
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event["routeArn"]
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                # The principal user identification associated with the token
                # sent by the client.
                "principalId": "abcdef",
                "policyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Action": "execute-api:Invoke",
                        "Effect": "Allow",
                        "Resource": event["routeArn"]
                    }]
                },
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
```

```

    }
    print('allowed')
    return response
else:
    print('denied')
    return response
except BaseException:
    print('denied')
    return response

```

Identitätsquellen

Sie können optional Identitätsquellen für einen Lambda-Genehmiger angeben. Identitätsquellen geben den Speicherort der Daten an, die für die Autorisierung einer Anforderung erforderlich sind. Beispielsweise können Sie Header- oder Abfragezeichenfolgenwerte als Identitätsquellen angeben. Wenn Sie Identitätsquellen angeben, müssen Clients diese in die Anforderung aufnehmen. Wenn die Anforderung des Clients die Identitätsquellen nicht enthält, ruft API Gateway Ihren Lambda-Genehmiger nicht auf, und der Client erhält einen 401-Fehler.

In der folgenden Tabelle werden die unterstützten Identitätsquellen für einen Lambda-Genehmiger beschrieben.

Typ	Beispiel	Hinweise
Header-Wert	<code>\$request.header. <i>name</i></code>	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.
Abfragezeichenfolgenwert	<code>\$request.querystring. <i>name</i></code>	Abfragezeichenfolgenamen unterscheiden zwischen Groß- und Kleinschreibung.
Kontextvariable	<code>\$kontext. <i>variableName</i></code>	Der Wert einer unterstützten Kontextvariablen .
Stufenvariable	<code>\$stageVariablen. <i>variableName</i></code>	Der Wert einer Stufenvariablen .

Sie können auch direkt `{"errorMessage" : "Unauthorized"}` aus Ihrer Lambda-Funktion zurückgeben, um einen 401-Fehler an die Clients zurückzugeben. Wenn Sie einen 401-Fehler direkt aus Ihrer Lambda-Funktion an die Clients zurückgeben, geben Sie beim Erstellen Ihres Lambda-Generators keine Identitätsquellen an.

Zwischenspeichern der Antworten des Generators

Sie können das Caching für einen Lambda-Autorisierer aktivieren, indem Sie einen `authorizerResultTtlInSeconds` angeben. Wenn für einen Generator Zwischenspeicherung aktiviert ist, verwendet API Gateway die Identitätsquellen des Generators als Cache-Schlüssel. Wenn ein Client dieselben Parameter in Identitätsquellen innerhalb der konfigurierten TTL angibt, verwendet API Gateway das zwischengespeicherte Generator-Ergebnis, anstatt Ihre Lambda-Funktion aufzurufen.

Um das Caching zu aktivieren, muss der Generator über mindestens eine Identitätsquelle verfügen.

Wenn Sie einfache Antworten für einen Generator aktivieren, erlaubt oder verweigert die Antwort des Generators alle API-Anforderungen, die mit den zwischengespeicherten Identitätsquellenwerten übereinstimmen, vollständig. Um detailliertere Berechtigungen zu erhalten, deaktivieren Sie einfache Antworten und geben Sie eine IAM-Richtlinie zurück. Je nach Generator muss Ihre IAM-Richtlinie möglicherweise den Zugriff auf mehrere Ressourcen steuern.

Standardmäßig verwendet API Gateway die zwischengespeicherte Generator-Antwort für alle Routen einer API, die den Generator verwenden. Um Antworten pro Route zwischenzuspeichern, fügen Sie den Identitätsquellen Ihres Generators `$context.routeKey` hinzu.

Erstellen eines Lambda-Generators

Wenn Sie einen Lambda-Generator erstellen, geben Sie die Lambda-Funktion an, die API Gateway verwenden soll. Sie müssen API Gateway die Berechtigung zum Aufrufen der Lambda-Funktion erteilen, indem Sie entweder die Ressourcenrichtlinie der Funktion oder eine IAM-Rolle verwenden. Der `create-authorizer`-Befehl erstellt einen Lambda-Generator:

```
aws apigatewayv2 create-authorizer \  
  --api-id abcdef123 \  
  --authorizer-type REQUEST \  
  --identity-source '$request.header.Authorization' \  
  --name lambda-authorizer \  
  --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/  
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \  
  --authorizer-payload-format-version '2.0' \  

```

```
--enable-simple-responses
```

Der folgende [add-permission](#)-Befehl aktualisiert die Ressourcenrichtlinie der Lambda-Funktion, um API Gateway die Berechtigung zum Aufrufen der Funktion zu erteilen. Wenn API Gateway nicht zum Aufrufen Ihrer Funktion berechtigt ist, erhalten Clients einen 500 Internal Server Error.

```
aws lambda add-permission \  
  --function-name my-authorizer-function \  
  --statement-id apigateway-invoke-permissions-abc123 \  
  --action lambda:InvokeFunction \  
  --principal apigateway.amazonaws.com \  
  --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-  
id/authorizers/authorizer-id"
```

Nachdem Sie einen Genehmiger erstellt und API Gateway die Berechtigung zum Aufrufen erteilt haben, aktualisieren Sie Ihre Route, um den Genehmiger zu verwenden. Der folgende [update-route](#)-Befehl fügt der Route den Lambda-Genehmiger hinzu. Wenn Ihr Lambda-Genehmiger eine Richtlinienpeicherung verwendet, stellen Sie sicher, dass Sie die Richtlinie aktualisieren, um den Zugriff für die zusätzliche Route zu steuern.

```
aws apigatewayv2 update-route \  
  --api-id abcdef123 \  
  --route-id abc123 \  
  --authorization-type CUSTOM \  
  --authorizer-id def123
```

Fehlerbehebung für Lambda-Genehmiger

Wenn API Gateway Ihren Lambda-Genehmiger nicht aufrufen kann oder Ihr Lambda-Genehmiger eine Antwort in einem ungültigen Format zurückgibt, erhalten Clients einen 500 Internal Server Error.

Um Fehler zu beheben, [aktivieren Sie die Zugriffsprotokollierung](#) für Ihre API-Stufe. Fügen Sie die `$context.authorizer.error`-Protokollierungsvariable in Ihr Protokollformat ein.

Wenn die Protokolle darauf hinweisen, dass API Gateway keine Berechtigung zum Aufrufen Ihrer Funktion besitzt, aktualisieren Sie die Ressourcenrichtlinie Ihrer Funktion oder stellen Sie eine IAM-Rolle bereit, um die API Gateway-Berechtigung zum Aufrufen des Genehmigers zu erteilen.

Wenn die Protokolle darauf hinweisen, dass Ihre Lambda-Funktion eine ungültige Antwort zurückgibt, überprüfen Sie, ob Ihre Lambda-Funktion eine Antwort im [erforderlichen Format](#) zurückgibt.

Steuern Sie den Zugriff auf HTTP APIs mit JWT-Autorisierern in API Gateway

Sie können JSON Web Tokens (JWTs) als Teil der Frameworks [OpenID Connect \(OIDC\)](#) und [OAuth 2.0](#) verwenden, um den Client-Zugriff auf Ihre zu beschränken. APIs

Wenn Sie einen JWT-Authorizer für eine Route Ihrer API konfigurieren, validiert API Gateway die, JWTs die Clients mit API-Anfragen einreichen. API Gateway genehmigt oder lehnt Anfragen basierend auf der Token-Validierung und optional Bereichen im Token ab. Wenn Sie Bereiche für eine Route konfigurieren, muss das Token mindestens einen der Bereiche der Route enthalten.

Sie können unterschiedliche Genehmiger für jede Route einer API konfigurieren oder denselben Genehmiger für mehrere Routen verwenden.

Note

Es gibt keinen Standardmechanismus, um JWT-Zugriffstoken von anderen Typen wie OpenID Connect ID-Tokens zu unterscheiden. Wir empfehlen, Ihre Routen so zu konfigurieren, dass Autorisierungsbereiche erforderlich sind, es sei denn, Sie benötigen ID-Token für die API-Autorisierung. Sie können Ihre JWT-Genehmiger auch so konfigurieren, dass Aussteller oder Zielgruppen erforderlich sind, die Ihr Identitätsanbieter nur bei der Ausgabe von JWT-Zugriffstoken verwendet.

Autorisieren von API-Anfragen mit einem JWT-Genehmiger

API Gateway verwendet den folgenden allgemeinen Workflow, um Anfragen an Routen zu autorisieren, die für die Verwendung eines JWT-Genheimigers konfiguriert sind.

1. Überprüfen Sie die [identitySource](#) auf ein Token. Der `identitySource` kann nur das Token oder das Token mit dem Präfix `Bearer` enthalten.
2. Decodieren Sie das Token.
3. Überprüfen Sie den Algorithmus und die Signatur des Tokens mit dem vom des Ausstellers abgerufenen öffentlichen Schlüsse `jwtks_uri`. Derzeit werden nur RSA-basierte Algorithmen unterstützt. API Gateway kann den öffentlichen Schlüssel zwei Stunden lang zwischenspeichern. Es hat sich bewährt, beim Wechseln von Schlüsseln eine bestimmte Frist einzuräumen, während der sowohl der alte als auch der neue Schlüssel gültig sind.
4. Validieren Sie die Ansprüche. API Gateway evaluiert die folgenden Token-Ansprüche:

- [kid](#) – Das Token muss einen Header-Anspruch haben, der mit dem Schlüssel in dem `jwt_header` übereinstimmt, der das Token signiert hat.
- [iss](#) – Muss mit dem für den Genehmiger konfigurierten [issuer](#) übereinstimmen.
- [aud](#) oder `client_id` – Muss mit einem der für den Genehmiger konfigurierten [audience](#)-Einträge übereinstimmen. API Gateway validiert `client_id` nur dann, wenn `aud` nicht vorhanden ist. Wenn sowohl `aud` als auch `client_id` vorhanden sind, bewertet API Gateway `aud`.
- [exp](#) – Muss nach der aktuellen Uhrzeit in UTC liegen.
- [nbf](#) – muss vor der aktuellen Uhrzeit in UTC liegen.
- [iat](#) – Muss vor der aktuellen Uhrzeit in UTC liegen.
- [scope](#) oder `scp` – Das Token muss mindestens einen der Bereiche in den [authorizationScopes](#) der Route enthalten.

Wenn einer dieser Schritte fehlschlägt, lehnt API Gateway die API-Anfrage ab.

Nach der Validierung des JWT übergibt API Gateway die Ansprüche im Token an die Integration der API-Route. Backend-Ressourcen, wie beispielsweise Lambda-Funktionen, können auf die JWT-Ansprüche zugreifen. Wenn das JWT beispielsweise den Identitätsanspruch `emailID` enthält, ist er für eine Lambda-Integration in `$event.requestContext.authorizer.jwt.claims.emailID` verfügbar. Weitere Informationen zur Nutzlast, die API Gateway an Lambda-Integrationen sendet, finden Sie unter [the section called “AWS Lambda Integrationen”](#).

Erstellen eines JWT-Genehmigers

Bevor Sie einen JWT-Genehmiger erstellen, müssen Sie eine Clientanwendung bei einem Identitätsanbieter registrieren. Sie müssen darüber hinaus eine HTTP-API erstellt haben. Beispiele zum Erstellen einer HTTP-API finden Sie unter [Erstellen einer HTTP-API](#).

Erstellen eines JWT-Genehmigers mit der Konsole

Die folgenden Schritte zeigen, wie Sie mit der Konsole einen JWT-Genehmiger erstellen.

So erstellen Sie mit der Konsole einen JWT-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine HTTP-API.

3. Wählen Sie im Hauptnavigationsbereich Autorisierung aus.
4. Wählen Sie die Registerkarte Genehmiger verwalten aus.
5. Wählen Sie Erstellen aus.
6. Wählen Sie unter Genehmiger-Typ den Wert JWT aus.
7. Konfigurieren Sie Ihren JWT-Genehmiger und geben Sie eine Identitätsquelle an, die die Quelle des Tokens definiert.
8. Wählen Sie Erstellen aus.

Erstellen Sie einen JWT-Autorisierer mit dem AWS CLI

Der folgende [create-authorizer](#)-Befehl erstellt einen JWT-Genehmiger. Bei `jwt-configuration` geben Sie das Audience und Issuer als Ihren Identitätsanbieter an. Wenn Sie Amazon Cognito als Identitätsanbieter verwenden, lautet die `IssuerUrl` `https://cognito-idp.us-east-2.amazonaws.com/userPoolID`.

```
aws apigatewayv2 create-authorizer \  
  --name authorizer-name \  
  --api-id api-id \  
  --authorizer-type JWT \  
  --identity-source '$request.header.Authorization' \  
  --jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Erstellen Sie einen JWT-Autorisierer mit AWS CloudFormation

Die folgende CloudFormation Vorlage erstellt eine HTTP-API mit einem JWT-Autorisierer, der Amazon Cognito als Identitätsanbieter verwendet.

Die Ausgabe der CloudFormation Vorlage ist eine URL für eine von Amazon Cognito gehostete Benutzeroberfläche, über die sich Kunden registrieren und anmelden können, um ein JWT zu erhalten. Nachdem sich ein Client angemeldet hat, wird er mit einem Zugriffstoken in der URL zu Ihrer HTTP-API weitergeleitet. Um die API mit dem Zugriffstoken aufzurufen, ändern Sie das # in der URL in ein ?, um das Token als Abfragezeichenfolgenparameter zu verwenden.

Beispielvorlage CloudFormation

```
AWSTemplateFormatVersion: '2010-09-09'  
Description: |  
  Example HTTP API with a JWT authorizer. This template includes an Amazon Cognito user  
  pool as the issuer for the JWT authorizer
```

and an Amazon Cognito app client as the audience for the authorizer. The outputs include a URL for an Amazon Cognito hosted UI where clients can sign up and sign in to receive a JWT. After a client signs in, the client is redirected to your HTTP API with an access token in the URL. To invoke the API with the access token, change the '#' in the URL to a '?' to use the token as a query string parameter.

Resources:

MyAPI:

```
Type: AWS::ApiGatewayV2::Api
Properties:
  Description: Example HTTP API
  Name: api-with-auth
  ProtocolType: HTTP
  Target: !GetAtt MyLambdaFunction.Arn
```

DefaultRouteOverrides:

```
Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
Properties:
  ApiId: !Ref MyAPI
  Route:
    AuthorizationType: JWT
    AuthorizerId: !Ref JWTAuthorizer
```

JWTAuthorizer:

```
Type: AWS::ApiGatewayV2::Authorizer
Properties:
  ApiId: !Ref MyAPI
  AuthorizerType: JWT
  IdentitySource:
    - '$request.querystring.access_token'
  JwtConfiguration:
    Audience:
      - !Ref AppClient
    Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}
  Name: test-jwt-authorizer
```

MyLambdaFunction:

```
Type: AWS::Lambda::Function
Properties:
  Runtime: nodejs18.x
  Role: !GetAtt FunctionExecutionRole.Arn
  Handler: index.handler
  Code:
    ZipFile: |
      exports.handler = async (event) => {
        const response = {
```

```

        statusCode: 200,
        body: JSON.stringify('Hello from the ' + event.routeKey + ' route!'),
    };
    return response;
};

APIInvokeLambdaPermission:
  Type: AWS::Lambda::Permission
  Properties:
    FunctionName: !Ref MyLambdaFunction
    Action: lambda:InvokeFunction
    Principal: apigateway.amazonaws.com
    SourceArn: !Sub arn:${AWS::Partition}:execute-api:${AWS::Region}:
${AWS::AccountId}:${MyAPI}/${default}/${default}
  FunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: http-api-user-pool
      AutoVerifiedAttributes:
        - email
      Schema:
        - Name: name
          AttributeDataType: String
          Mutable: true
          Required: true
        - Name: email
          AttributeDataType: String
          Mutable: false
          Required: true
  AppClient:
    Type: AWS::Cognito::UserPoolClient

```

```

Properties:
  Allowed0AuthFlows:
    - implicit
  Allowed0AuthScopes:
    - aws.cognito.signin.user.admin
    - email
    - openid
    - profile
  Allowed0AuthFlowsUserPoolClient: true
  ClientName: api-app-client
  CallbackURLs:
    - !Sub https://${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
  ExplicitAuthFlows:
    - ALLOW_USER_PASSWORD_AUTH
    - ALLOW_REFRESH_TOKEN_AUTH
  UserPoolId: !Ref UserPool
  SupportedIdentityProviders:
    - COGNITO
HostedUI:
  Type: AWS::Cognito::UserPoolDomain
  Properties:
    Domain: !Join
      - '-'
      - - !Ref MyAPI
        - !Ref AppClient
    UserPoolId: !Ref UserPool
Outputs:
  SignupURL:
    Value: !Sub https://${HostedUI}.auth.${AWS::Region}.amazoncognito.com/login?
client_id=${AppClient}&response_type=token&scope=email+profile&redirect_uri=https://
${MyAPI}.execute-api.${AWS::Region}.amazonaws.com

```

Aktualisieren einer Route zum Verwenden eines JWT-Genehmigers

Sie können die Konsole, das oder ein AWS SDK verwenden AWS CLI, um eine Route so zu aktualisieren, dass sie einen JWT-Autorisierer verwendet.

Aktualisieren einer Route zum Verwenden eines JWT-Genehmigers mit der Konsole

Die folgenden Schritte zeigen, wie Sie mit der Konsole eine Route so aktualisieren, dass sie einen JWT-Genehmiger verwendet.

So erstellen Sie mit der Konsole einen JWT-Genehmiger

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie im Hauptnavigationsbereich Autorisierung aus.
4. Wählen Sie zunächst eine Methode und anschließend Ihren Genehmiger aus dem Dropdown-Menü aus und klicken Sie auf Genehmiger anfügen.

Aktualisieren Sie eine Route, um einen JWT-Autorisierer zu verwenden, indem Sie AWS CLI

Mit dem folgenden [update-route](#)-Befehl wird eine Route so aktualisiert, dass sie einen JWT-Genehmiger verwendet.

```
aws apigatewayv2 update-route \  
  --api-id api-id \  
  --route-id route-id \  
  --authorization-type JWT \  
  --authorizer-id authorizer-id \  
  --authorization-scopes user.email
```

Steuern Sie den Zugriff auf HTTP APIs mit der IAM-Autorisierung in API Gateway

Sie können die IAM-Autorisierung für HTTP-API-Routen aktivieren. Wenn die IAM-Autorisierung aktiviert ist, müssen Clients [Signature Version 4 \(Sigv4\)](#) verwenden, um ihre Anfragen mit AWS Anmeldeinformationen zu signieren. API Gateway ruft Ihre API-Route nur auf, wenn der Client über die `execute-api`-Berechtigung für die Route verfügt.

[Die IAM-Autorisierung für HTTP APIs ähnelt der für REST. APIs](#)

Note

Ressourcenrichtlinien werden derzeit für HTTP APIs nicht unterstützt.

Beispiele für IAM-Richtlinien, die Clients die Berechtigung zum Aufrufen gewähren APIs, finden Sie unter [the section called “Kontrollieren des Zugriffs für den API-Aufruf”](#)

Aktivieren der IAM-Autorisierung für eine Route

Der folgende [update-route](#)-Befehl aktiviert die IAM-Autorisierung für eine HTTP-API-Route.

```
aws apigatewayv2 update-route \  
  --api-id abc123 \  
  --route-id abcdef \  
  --authorization-type AWS_IAM
```

Erstellen von Integrationen für HTTP-APIs in API Gateway

Integrationen verbinden eine Route mit Backend-Ressourcen. HTTP-APIs unterstützen Lambda-Proxy-, AWS-Service- und HTTP-Proxy-Integrationen. Beispielsweise können Sie eine POST-Anfrage an die `/signup`-Route Ihrer API so konfigurieren, dass sie in eine Lambda-Funktion integriert wird, die die Registrierung von Kunden übernimmt.

Themen

- [Erstellen Sie AWS Lambda Proxy-Integrationen für HTTP APIs in API Gateway](#)
- [Erstellen Sie HTTP-Proxyintegrationen für HTTP APIs](#)
- [Erstellen Sie AWS Dienstintegrationen für HTTP APIs in API Gateway](#)
- [Erstellen Sie private Integrationen für HTTP APIs in API Gateway](#)

Erstellen Sie AWS Lambda Proxy-Integrationen für HTTP APIs in API Gateway

Eine Lambda-Proxy-Integration ermöglicht es Ihnen, eine API-Route in eine Lambda-Funktion zu integrieren. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion und gibt die Antwort der Funktion an den Client zurück. Beispiele zum Erstellen einer HTTP-API finden Sie unter [Erstellen einer HTTP-API](#).

Nutzlastformatversion

Die Nutzlastformatversion gibt das Format des Ereignisses an, das API Gateway an eine Lambda-Integration sendet und wie API Gateway die Lambda-Antwort interpretiert. Wenn Sie keine Version im Payload-Format angeben, AWS-Managementkonsole wird standardmäßig die neueste Version verwendet. Wenn Sie eine Lambda-Integration mithilfe des AWS CLI, CloudFormation, oder eines SDK erstellen, müssen Sie eine `payloadFormatVersion` angeben. Die unterstützten Werte sind `1.0` und `2.0`.

Weitere Informationen zum Einrichten von `payloadFormatVersion`, finden Sie unter [create-integration](#). Weitere Informationen zur Ermittlung des `payloadFormatVersion` einer vorhandenen Integration finden Sie unter [get-integration](#)

Nutzlastformatunterschiede

In der folgenden Liste sehen Sie die Unterschiede zwischen den Payload-Formatversionen 1.0 und 2.0:

- Das Format 2.0 hat keine `multiValueHeaders`- oder `multiValueQueryStringParameters`-Felder. Doppelte Header werden mit Kommas kombiniert und in das `headers`-Feld aufgenommen. Doppelte Abfragezeichenfolgen werden mit Kommas kombiniert und in das `queryStringParameters`-Feld aufgenommen.
- Das Format 2.0 hat `rawPath`. Wenn Sie eine API-Zuordnung verwenden, um Ihre Stufe mit einem benutzerdefinierten Domainnamen zu verbinden, stellt `rawPath` den API-Zuordnungswert nicht bereit. Verwenden Sie die Formate 1.0 und `path`, um auf die API-Zuordnung für Ihren benutzerdefinierten Domainnamen zuzugreifen.
- Das Format 2.0 enthält ein neues `cookies`-Feld. Alle Cookie-Header in der Anforderung werden mit Kommas kombiniert und dem `cookies`-Feld hinzugefügt. In der Antwort an den Client wird jedes Cookie zu einem `set-cookie`-Header.

Nutzdatenformatstruktur

Die folgenden Beispiele zeigen die Struktur jeder Nutzlastformatversion. Alle Header-Namen werden in Kleinbuchstaben angegeben.

2.0

```
{
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": [
    "cookie1",
    "cookie2"
  ],
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
```

```

},
"queryStringParameters": {
  "parameter1": "value1,value2",
  "parameter2": "value"
},
"requestContext": {
  "accountId": "123456789012",
  "apiId": "api-id",
  "authentication": {
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  }
},
"authorizer": {
  "jwt": {
    "claims": {
      "claim1": "value1",
      "claim2": "value2"
    },
    "scopes": [
      "scope1",
      "scope2"
    ]
  }
},
"domainName": "id.execute-api.us-east-1.amazonaws.com",
"domainPrefix": "id",
"http": {
  "method": "POST",
  "path": "/my/path",
  "protocol": "HTTP/1.1",
  "sourceIp": "192.0.2.1",
  "userAgent": "agent"
},
"requestId": "id",
"routeKey": "$default",
"stage": "$default",

```

```
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "body": "Hello from Lambda",
  "pathParameters": {
    "parameter1": "value1"
  },
  "isBase64Encoded": false,
  "stageVariables": {
    "stageVariable1": "value1",
    "stageVariable2": "value2"
  }
}
```

1.0

```
{
  "version": "1.0",
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],

```

```
"parameter2": [
  "value"
],
"requestContext": {
  "accountId": "123456789012",
  "apiId": "id",
  "authorizer": {
    "claims": null,
    "scopes": null
  },
  "domainName": "id.execute-api.us-east-1.amazonaws.com",
  "domainPrefix": "id",
  "extendedRequestId": "request-id",
  "httpMethod": "GET",
  "identity": {
    "accessKey": null,
    "accountId": null,
    "caller": null,
    "cognitoAuthenticationProvider": null,
    "cognitoAuthenticationType": null,
    "cognitoIdentityId": null,
    "cognitoIdentityPoolId": null,
    "principalOrgId": null,
    "sourceIp": "192.0.2.1",
    "user": null,
    "userAgent": "user-agent",
    "userArn": null,
    "clientCert": {
      "clientCertPem": "CERT_CONTENT",
      "subjectDN": "www.example.com",
      "issuerDN": "Example issuer",
      "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
      "validity": {
        "notBefore": "May 28 12:30:02 2019 GMT",
        "notAfter": "Aug  5 09:36:04 2021 GMT"
      }
    }
  },
  "path": "/my/path",
  "protocol": "HTTP/1.1",
  "requestId": "id=",
  "requestTime": "04/Mar/2020:19:15:17 +0000",
  "requestTimeEpoch": 1583349317135,
```

```
"resourceId": null,
"resourcePath": "/my/path",
"stage": "$default"
},
"pathParameters": null,
"stageVariables": null,
"body": "Hello from Lambda!",
"isBase64Encoded": false
}
```

Antwortformat der Lambda-Funktion

Die Nutzlastformatversion bestimmt die Struktur der Antwort, die Ihre Lambda-Funktion zurückgeben muss.

Lambda-Funktionsantwort für Format 1.0

Bei der 1.0-Formatversion müssen Lambda-Integrationen eine Antwort im folgenden JSON-Format zurückgeben.

Example

```
{
  "isBase64Encoded": true|false,
  "statusCode": httpStatusCode,
  "headers": { "headername": "headervalue", ... },
  "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
  "body": "..."
}
```

Lambda-Funktionsantwort für Format 2.0

Bei der 2.0-Formatversion kann API Gateway das Antwortformat für Sie ableiten. API Gateway geht von den folgenden Annahmen aus, wenn Ihre Lambda-Funktion gültigen JSON-Code und keinen zurückgib `statusCode`:

- `isBase64Encoded` ist `false`.
- `statusCode` ist `200`.
- `content-type` ist `application/json`.
- `body` ist die Antwort der Funktion.

Die folgenden Beispiele zeigen die Ausgabe einer Lambda-Funktion und die Interpretation von API Gateway.

Lambda-Funktionsausgabe	API Gateway-Interpretation
<pre>"Hello from Lambda!"</pre>	<pre>{ "isBase64Encoded": false, "statusCode": 200, "body": "Hello from Lambda!", "headers": { "content-type": "application/ json" } }</pre>
<pre>{ "message": "Hello from Lambda!" }</pre>	<pre>{ "isBase64Encoded": false, "statusCode": 200, "body": "{ \"message\": \"Hello from Lambda!\" }", "headers": { "content-type": "application/ json" } }</pre>

Um die Antwort anzupassen, sollte Ihre Lambda-Funktion eine Antwort im folgenden Format zurückgeben.

```
{
  "cookies" : ["cookie1", "cookie2"],
  "isBase64Encoded": true|false,
  "statusCode": httpStatusCode,
  "headers": { "headername": "headervalue", ... },
  "body": "Hello from Lambda!"
}
```


Erstellen Sie HTTP-Proxyintegrationen für HTTP APIs

Eine HTTP-Proxy-Integration ermöglicht es Ihnen, eine API-Route mit einem öffentlich routingfähigen HTTP-Endpunkt zu verbinden. Bei diesem Integrationstyp wird die gesamte Anforderung und Antwort von API Gateway zwischen dem Frontend und dem Backend übergeben.

Geben Sie die URL eines öffentlich routingfähigen HTTP-Endpunkts an, um eine HTTP-Proxy-Integration zu erstellen.

HTTP-Proxy-Integration mit Pfadvariablen

Sie können Pfadvariablen in HTTP-API-Routen verwenden.

Beispielsweise fängt die Route `/pets/{petID}` Anforderungen an `/pets/6` ab. Sie können Pfadvariablen im Integrations-URI referenzieren, um den Inhalt der Variablen an eine Integration zu senden. Ein Beispiel ist `/pets/extendedpath/{petID}`.

Sie können gierige Pfadvariablen verwenden, um alle untergeordneten Ressourcen einer Route zu erfassen. Um eine gierige Pfadvariable zu erstellen, fügen Sie `+` dem Variablennamen hinzu, z. B. `{proxy+}`.

Um eine Route mit einer HTTP-Proxy-Integration einzurichten, die alle Anforderungen abfängt, erstellen Sie eine API-Route mit einer gierigen Pfadvariable (z. B. `/parent/{proxy+}`). Integrieren Sie die Route mit einem HTTP-Endpunkt (z. B. `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) für die ANY-Methode. Die gierige Pfadvariable muss am Ende des Ressourcenpfads stehen.

Erstellen Sie AWS Dienstintegrationen für HTTP APIs in API Gateway

Sie können Ihre HTTP-API mithilfe erstklassiger Integrationen in AWS Dienste integrieren. Eine erstklassige Integration verbindet eine HTTP-API-Route mit einer AWS -Service-API. Wenn ein Client eine Route aufruft, die durch eine erstklassige Integration unterstützt wird, ruft API Gateway eine AWS Service-API für Sie auf. Sie können beispielsweise erstklassige Integrationen verwenden, um eine Nachricht an eine Amazon Simple Queue Service-Warteschlange zu senden oder eine AWS Step Functions Zustandsmaschine zu starten. Unterstützte Serviceaktionen finden Sie unter [the section called “AWSReferenz für -Serviceintegrationen”](#).

Mapping von Anforderungsparametern

Erstklassige Integrationen verfügen über erforderliche und optionale Parameter. Sie müssen alle erforderlichen Parameter zum Erstellen einer Integration konfigurieren. Sie können statische

Werte oder Mapping-Parameter verwenden, die zur Laufzeit dynamisch ausgewertet werden. Eine vollständige Liste der unterstützten Integrationen und Parameter finden Sie unter [the section called “AWSReferenz für -Serviceintegrationen”](#).

In der Tabelle unten sind die unterstützten Anforderungsparameter der Zuordnung beschrieben.

Typ	Beispiel	Hinweise
Header-Wert	<code>\$request.header. <i>name</i></code>	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. "header1": "value1,value2" .
Abfragezeichenfolgenwert	<code>\$request.querystring. <i>name</i></code>	Abfragezeichenfolgennamen unterscheiden zwischen Groß- und Kleinschreibung. API Gateway kombiniert mehrere Werte mit Kommas, z. B. "querystring1": "Value1,Value2" .
Path-Parameter	<code>\$request.path. <i>name</i></code>	Der Wert eines Pfadparameters in der Anforderung. Beispiel: Wenn die Route <code>/pets/{petId}</code> ist, können Sie den Parameter <code>petId</code> aus der Anforderung mit <code><i>\$request.path.petId</i></code> zuordnen.
Übergeben des Anforderungstexts	<code>\$request.body</code>	API Gateway übergibt den gesamten Anforderungstext.
Anforderungstext	<code>\$request.body. <i>name</i></code>	Ein JSON-Pfadausdruck . Rekursiver Abstieg

Typ	Beispiel	Hinweise
		<code>(\$request.body.. <i>name</i>)</code> und Filterausdrücke (<code>(<i>expression</i>)</code>) werden nicht unterstützt.  Note Wenn Sie einen JSON-Pfad angeben, schneidet API Gateway den Anfragetext auf 100 KB ab und wendet dann den Auswahl Ausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an <code>\$request.body</code> .
Kontextvariable	<code>\$kontext. <i>variableName</i></code>	Der Wert einer unterstützten Kontextvariablen .
Stufenvariable	<code>\$stageVariablen. <i>variableName</i></code>	Der Wert einer Stufenvariablen .
Statischer Wert	<code><i>string</i></code>	Ein konstanter Wert.

Erstellen Sie einer erstklassigen Integration

Bevor Sie eine erstklassige Integration erstellen, müssen Sie eine IAM-Rolle erstellen, die API Gateway Gateway-Berechtigungen zum Aufrufen der AWS Serviceaktion gewährt, mit der Sie die Integration durchführen. Weitere Informationen finden Sie unter [Erstellen einer Rolle für einen AWS - Service](#).

Um eine erstklassige Integration zu erstellen, wählen Sie eine unterstützte AWS Serviceaktion aus, z. B. SQS-SendMessage konfigurieren Sie die Anforderungsparameter, und stellen Sie eine Rolle bereit, die API Gateway Gateway-Berechtigungen zum Aufrufen der integrierten AWS Service-API gewährt. Je nach Integration-Subtyp sind unterschiedliche Anforderungsparameter erforderlich. Weitere Informationen hierzu finden Sie unter [the section called “AWSReferenz für - Serviceintegrationen”](#).

Der folgende [create-integration](#)-Befehl erstellt eine Integration, die eine Amazon-SQS-Nachricht sendet:

```
aws apigatewayv2 create-integration \
  --api-id abcdef123 \
  --integration-subtype SQS-SendMessage \
  --integration-type AWS_PROXY \
  --payload-format-version 1.0 \
  --credentials-arn arn:aws:iam::123456789012:role/apigateway-sqs \
  --request-parameters '{"QueueUrl": "$request.header.queueUrl", "MessageBody":
"$request.body.message"}'
```

Erstellen Sie eine erstklassige Integration mit CloudFormation

Das folgende Beispiel zeigt ein CloudFormation Snippet, das eine `/ {source} / {detailType}` Route mit einer erstklassigen Integration mit Amazon erstellt. EventBridge

Der Parameter `Source` wird dem Pfadparameter `{source}`, der `DetailType` dem Pfadparameter `{DetailType}` und der Parameter `Detail` dem Anforderungstext zugeordnet.

Das Snippet zeigt weder den Event Bus noch die IAM-Rolle, die API-Gateway-Berechtigungen zum Aufrufen der Aktion `PutEvents` gewährt.

```
Route:
  Type: AWS::ApiGatewayV2::Route
  Properties:
    ApiId: !Ref HttpApi
    AuthorizationType: None
    RouteKey: 'POST / {source} / {detailType}'
    Target: !Join
      - /
      - - integrations
        - !Ref Integration
  Integration:
```

```
Type: AWS::ApiGatewayV2::Integration
Properties:
  ApiId: !Ref HttpApi
  IntegrationType: AWS_PROXY
  IntegrationSubtype: EventBridge-PutEvents
  CredentialsArn: !GetAtt EventBridgeRole.Arn
  RequestParameters:
    Source: $request.path.source
    DetailType: $request.path.detailType
    Detail: $request.body
    EventBusName: !GetAtt EventBus.Arn
  PayloadFormatVersion: "1.0"
```

Integration-Subtypen

Die folgenden [Integration-Subtypen](#) werden für HTTP-APIs unterstützt.

Integration-Subtypen

- [EventBridge-PutEvents 1.0](#)
- [SQS-SendMessage 1.0](#)
- [SQS-ReceiveMessage 1.0](#)
- [SQS-DeleteMessage 1.0](#)
- [SQS-PurgeQueue 1.0](#)
- [AppConfig-GetConfiguration 1.0](#)
- [Kinesis-PutRecord 1.0](#)
- [StepFunctions-StartExecution 1.0](#)
- [StepFunctions-StartSyncExecution 1.0](#)
- [StepFunctions-StopExecution 1.0](#)

EventBridge-PutEvents 1.0

Sendet benutzerdefinierte Ereignisse an Amazon EventBridge, damit sie Regeln zugeordnet werden können.

Parameter	Erforderlich
Detail	Wahr

Parameter	Erforderlich
DetailType	Wahr
Source	Wahr
Zeit	Falsch
EventBusName	Falsch
Ressourcen	Falsch
Region	Falsch
TraceHeader	Falsch

Weitere Informationen finden Sie unter [PutEvents](#) in der Amazon EventBridge-API-Referenz.

SQS-SendMessage 1.0

Übergibt eine Nachricht an die angegebene Warteschlange

Parameter	Erforderlich
QueueUrl	Wahr
MessageBody	Wahr
DelaySeconds	Falsch
MessageAttributes	Falsch
MessageDeduplicationId	Falsch
MessageGroupId	Falsch
MessageSystemAttributes	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [SendMessage](#) in der Amazon Simple Queue Service – API-Referenz.

SQS-ReceiveMessage 1.0

Ruft eine oder mehrere Nachrichten (bis zu 10) aus der angegebenen Warteschlange ab.

Parameter	Erforderlich
QueueUrl	Wahr
AttributeNames	Falsch
MaxNumberOfMessages	Falsch
MessageAttributeNames	Falsch
ReceiveRequestAttemptId	Falsch
VisibilityTimeout	Falsch
WaitTimeSeconds	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [ReceiveMessage](#) in der Amazon Simple Queue Service – API-Referenz.

SQS-DeleteMessage 1.0

Löscht die angegebene Nachricht aus der angegebenen Warteschlange

Parameter	Erforderlich
ReceiptHandle	Wahr
QueueUrl	Wahr
Region	Falsch

Weitere Informationen finden Sie unter [DeleteMessage](#) in der Amazon Simple Queue Service – API-Referenz.

SQS-PurgeQueue 1.0

Löscht alle Nachrichten in der angegebenen Warteschlange.

Parameter	Erforderlich
QueueUrl	Wahr
Region	Falsch

Weitere Informationen finden Sie unter [PurgeQueue](#) in der Amazon Simple Queue Service – API-Referenz.

AppConfig-GetConfiguration 1.0

Fordert Informationen zu einer Konfiguration an.

Parameter	Erforderlich
Anwendung	Wahr
Umgebung	Wahr
Konfiguration	Wahr
ClientId	Wahr
ClientConfigurationVersion	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [GetConfiguration](#) in der AWS-AppConfig-API-Referenz.

Kinesis-PutRecord 1.0

Schreibt einen einzelnen Datensatz in einen Amazon Kinesis-Daten-Stream.

Parameter	Erforderlich
StreamName	Wahr
Daten	Wahr
PartitionKey	Wahr
SequenceNumberForOrdering	Falsch
ExplicitHashKey	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [PutRecord](#) in der Amazon Kinesis Data Streams – API-Referenz.

StepFunctions-StartExecution 1.0

Startet eine Ausführung des Zustandsautomaten.

Parameter	Erforderlich
StateMachineArn	Wahr
Name	Falsch
Eingabe	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [StartExecution](#) in der AWS Step Functions-API-Referenz.

StepFunctions-StartSyncExecution 1.0

Startet eine synchrone Ausführung des Zustandsautomaten.

Parameter	Erforderlich
StateMachineArn	Wahr
Name	Falsch
Eingabe	Falsch
Region	Falsch
TraceHeader	Falsch

Weitere Informationen finden Sie unter [StartSyncExecution](#) in der AWS Step Functions-API-Referenz.

StepFunctions-StopExecution 1.0

Stoppt eine Ausführung.

Parameter	Erforderlich
ExecutionArn	Wahr
Ursache	Falsch
Fehler	Falsch
Region	Falsch

Weitere Informationen finden Sie unter [StopExecution](#) in der AWS Step Functions-API-Referenz.

Erstellen Sie private Integrationen für HTTP APIs in API Gateway

Private Integrationen ermöglichen es Ihnen, API-Integrationen mit privaten Ressourcen in einer VPC zu erstellen, z. B. Application Load Balancer oder containerbasierte Amazon ECS-Anwendungen.

Sie können Ihre Ressourcen in einer VPC für den Zugriff von Clients außerhalb der VPC bereitstellen, indem Sie private Integrationen verwenden. Sie können den Zugriff auf Ihre API steuern, indem Sie eine der von API Gateway unterstützten [Autorisierungsmethoden](#) verwenden.

 Note

Um eine private Integration zu erstellen, müssen Sie zunächst einen VPC-Link erstellen. VPC-Links V2 werden jetzt sowohl für HTTP als auch für REST APIs unterstützt. Weitere Informationen zu VPC-Links V2 finden Sie unter [the section called “VPC-Verbindungen V2”](#).

Nachdem Sie einen VPC-Link V2 erstellt haben, können Sie private Integrationen einrichten, die eine Verbindung zu einem Application Load Balancer, Network Load Balancer oder Ressourcen herstellen, die bei einem Service registriert sind. AWS Cloud Map

Überlegungen

Die folgenden Überlegungen könnten sich auf Ihre Verwendung privater Integrationen auswirken:

- Alle Ressourcen müssen sich im Besitz derselben AWS-Konto Person befinden. Dazu gehören der Load Balancer oder AWS Cloud Map Service, der VPC-Link und die HTTP-API.
- Standardmäßig verwendet der private Integrationsdatenverkehr das HTTP-Protokoll. Um HTTPS zu verwenden, geben Sie an. [tlsConfig](#) Um dies mithilfe von zu tun AWS-Managementkonsole, wählen Sie bei der Erstellung Ihrer privaten Integration Erweiterte Einstellungen und geben Sie dann einen sicheren Servernamen ein.
- Bei privaten Integrationen schließt API Gateway den [Stufen](#)-Teil des API-Endpunkts in die Anforderung an Ihre Backend-Ressourcen ein. Beispielsweise enthält eine Anforderung an die Stufe test einer API in der Anforderung an Ihre private Integration test/*route-path*. Um den Künstlernamen aus der Anforderung an Ihre Back-End-Ressourcen zu entfernen, verwenden Sie [Parameterzuordnung](#), um den Anforderungspfad nach `$request.path` zu überschreiben.

Erstellen einer privaten Integration mit einem Application Load Balancer oder Network Load Balancer

Bevor Sie eine private Integration erstellen, müssen Sie einen VPC-Link V2 erstellen. Weitere Informationen zu VPC-Links V2 finden Sie unter [the section called “VPC-Verbindungen V2”](#).

Um eine private Integration mit einem Application Load Balancer oder Network Load Balancer zu erstellen, erstellen Sie eine HTTP-Proxy-Integration, geben Sie den zu verwendenden VPC-Link an und stellen Sie den Listener-ARN des Load Balancers bereit.

Der folgende [create-integration](#)-Befehl erstellt eine private Integration, die über einen VPC-Link eine Verbindung mit einem Load Balancer herstellt.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
  --integration-method GET --connection-type VPC_LINK \
  --connection-id VPC-link-ID \
  --integration-uri arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/  
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65  
--payload-format-version 1.0
```

Erstellen Sie mithilfe von AWS Cloud Map Service Discovery eine private Integration

Bevor Sie eine private Integration erstellen, müssen Sie einen VPC-Link V2 erstellen. Weitere Informationen zu VPC-Links finden Sie unter [the section called “VPC-Verbindungen V2”](#).

Bei Integrationen mit AWS Cloud Map verwendet DiscoverInstances API Gateway Ressourcen. Sie können Abfrageparameter verwenden, um bestimmte Ressourcen zu adressieren. Die Attribute der registrierten Ressourcen müssen IP-Adressen und Ports enthalten. API Gateway verteilt Anforderungen auf fehlerfreie Ressourcen, die von zurückgegeben werde DiscoverInstances. Weitere Informationen finden Sie [DiscoverInstances](#) in der AWS Cloud Map API-Referenz.

 Note

Wenn Sie Amazon ECS zum Ausfüllen von Einträgen verwenden AWS Cloud Map, müssen Sie Ihre Amazon ECS-Aufgabe so konfigurieren, dass sie SRV-Datensätze mit Amazon ECS Service Discovery verwendet, oder Amazon ECS Service Connect aktivieren. Weitere Informationen finden Sie unter [Services miteinander verbinden](#) im Amazon Elastic Container Service-Entwicklerhandbuch.

Um eine private Integration mit zu erstellen AWS Cloud Map, erstellen Sie eine HTTP-Proxyintegration, geben Sie den zu verwendenden VPC-Link an und geben Sie den ARN des AWS Cloud Map Dienstes an.

Der folgende Befehl [create-integration](#) erstellt eine private Integration, die AWS Cloud Map Service Discovery verwendet, um Ressourcen zu identifizieren:

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
  --integration-method GET --connection-type VPC_LINK \
  --connection-id VPC-link-ID \
  --integration-uri arn:aws:servicediscovery:us-east-2:123456789012:service/srv-id?  
stage=prod&deployment=green_deployment
```

```
--payload-format-version 1.0
```

CORS für HTTP APIs im API Gateway konfigurieren

[Cross-Origin Resource Sharing \(CORS\)](#) ist eine Browser-Sicherheitsfunktion, die HTTP-Anfragen einschränkt, die von im Browser ausgeführten Skripten eingeleitet werden. Wenn Sie nicht auf Ihre API zugreifen können und eine Fehlermeldung erhalten, die `Cross-Origin Request Blocked` enthält, müssen Sie möglicherweise CORS aktivieren. Weitere Informationen finden Sie unter [Was ist CORS?](#).

CORS ist in der Regel erforderlich, um Webanwendungen zu erstellen, die auf einer anderen Domain oder einem anderen Ursprung APIs gehostet werden. Sie können CORS aktivieren, um Anfragen an Ihre API von einer Webanwendung zuzulassen, die auf einer anderen Domäne gehostet wird. Wenn Ihre API beispielsweise auf `https://{api_id}.execute-api.{region}.amazonaws.com/` gehostet wird und Sie Ihre API über eine Webanwendung aufrufen möchten, die auf `example.com` gehostet wird, muss Ihre API CORS unterstützen.

Wenn Sie CORS für eine API konfigurieren, sendet API Gateway automatisch eine Antwort auf Preflight-OPTIONS-Anfragen, selbst wenn keine OPTIONS-Route für Ihre API konfiguriert ist. Bei einer CORS-Anfrage fügt API Gateway die konfigurierten CORS-Header zur Antwort einer Integration hinzu.

Note

Wenn Sie CORS für eine API konfigurieren, ignoriert API Gateway CORS-Header, die von Ihrer Backend-Integration zurückgegeben werden.

Sie können die folgenden Parameter in einer CORS-Konfiguration angeben. Um diese Parameter mithilfe der HTTP-API-Konsole des API Gateway hinzuzufügen, wählen Sie **Add (Hinzufügen)**, nachdem Sie Ihren Wert eingegeben haben.

CORS-Header	CORS-Konfigurationseigenschaft	Beispielwerte
Access-Control-Allow-Origin	allowOrigins	<code>https://www.example.com</code>

CORS-Header	CORS-Konfigurationseigenschaft	Beispielwerte
		• * (alle Ursprünge zulassen) • https://* (einen beliebigen Ursprung zulassen, der mit https:// beginnt) • http://* (einen beliebigen Ursprung zulassen, der mit http:// beginnt)
Access-Control-Allow-Credentials	allowCredentials	true
Access-Control-Expose-Headers	exposeHeaders	Datum, * x-api-id
Access-Control-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	GET, POST, DELETE, *
Access-Control-Allow-Headers	allowHeaders	authorization, *

Um CORS-Header zurückzugeben, muss Ihre Anfrage einen `origin`-Header enthalten. Für die `OPTIONS`-Methode muss Ihre Anfrage einen `origin`-Header und einen `Access-Control-Request-Method`-Header enthalten.

Ihre CORS-Konfiguration sieht möglicherweise wie folgt aus:

API Gateway <

APIs

Custom domain names

VPC links

API: HelloWorld...(abcd1234)

▼ **Develop**

Routes

Authorization

Integrations

CORS

Reimport

Export

▼ **Deploy**

Stages

▼ **Monitor**

Metrics

Logging

▼ **Protect**

Throttling

[API Gateway](#) > [APIs](#) > [HelloWorld \(abcd1234\)](#) > **CORS**

Cross-Origin Resource Sharing

Configure CORS [Info](#)

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our [CORS documentation](#) for more details.

Access-Control-Allow-Origin

Enter a value for Allowed Origins

[Add](#)

https://www.example.com [×](#)

Access-Control-Allow-Headers

Enter a value for Allowed Headers

[Add](#)

authorization [×](#)

Access-Control-Allow-Methods

Choose Allowed Methods ▼

* [×](#)

Access-Control-Expose-Headers

Enter a value for Exposed Headers

[Add](#)

date,x-api-id [×](#)

Access-Control-Max-Age

300

Access-Control-Allow-Credentials

☒ YES

[Cancel](#)
[Save](#)

Konfigurieren von CORS für eine HTTP-API mit einer `$default`-Route und einem Genehmiger

Sie können CORS aktivieren und die Autorisierung für jede Route einer HTTP-API konfigurieren. Wenn Sie CORS und die Autorisierung für die `$default`-Route aktivieren, gibt es einige besondere Überlegungen. Die `$default`-Route fängt Anforderungen für alle Methoden und Routen ab, die Sie nicht explizit definiert haben, einschließlich OPTIONS-Anforderungen. Fügen Sie Ihrer API zur Unterstützung von nicht autorisierten OPTIONS-Anforderungen eine `OPTIONS /{proxy+}`-Route hinzu, die keine Autorisierung erfordert. Die Route `OPTIONS /{proxy+}` hat eine höhere Priorität als die Route `$default`. Dadurch können Clients ohne Autorisierung OPTIONS-Anforderungen an Ihre API senden. Weitere Informationen zu Weiterleitungsprioritäten finden Sie unter [Weiterleiten von API-Anforderungen](#).

Konfigurieren Sie CORS für eine HTTP-API mithilfe der CLI AWS

Der folgende `update-api`-Befehl aktiviert CORS-Anfragen von `https://www.example.com`:

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://www.example.com"
```

Weitere Informationen finden Sie unter [CORS](#) in der API-Referenz für Amazon API Gateway Version 2.

API-Anforderungen und -Antworten für HTTP-APIs in API Gateway transformieren

Sie können API-Anforderungen von Clients ändern, bevor sie Ihre Backend-Integrationen erreichen. Sie können auch die Antwort von Integrationen ändern, bevor API Gateway die Antwort an Clients zurückgibt. Sie verwenden das Parameter-Mapping, um API-Anforderungen und -Antworten für HTTP-APIs zu ändern. Um das Parameter-Mapping zu verwenden, geben Sie zu ändernde API-Anforderungs- oder Antwortparameter an und wie diese Parameter geändert werden sollen.

Transformieren von API-Anforderungen

Sie verwenden Anforderungsparameter, um Anforderungen zu ändern, bevor sie Ihre Backend-Integrationen erreichen. Sie können Header, Abfragezeichenfolgen oder den Anforderungspfad ändern.

Anforderungsparameter sind eine Schlüssel-Wert-Zuordnung. Der Schlüssel gibt den Speicherort des zu ändernden Anforderungsparameters und an, wie er geändert werden soll. Der Wert gibt die neuen Daten für den Parameter an.

Die folgende Tabelle zeigt unterstützte Schlüssel.

Typ	Syntax
Header	append overwrite remove:header. <i>headername</i>
Abfragezeichenfolge	append overwrite remove:querystring. <i>querystring-name</i>
Pfad	overwrite:path

Die folgende Tabelle zeigt unterstützte Werte, die Sie Parametern zuordnen können.

Typ	Syntax	Hinweise
Header-Wert	<code>\$request.header.<i>name</i></code> oder <code>\${request.header.<i>name</i>}</code>	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. "header1" : "value1,value2" . Einige Header sind reserviert. Weitere Informationen hierzu finden Sie unter the section called "Reservierte Header" .
Abfragezeichenfolgenwert	<code>\$request.querystring.<i>name</i></code> oder <code>\${request.querystring.<i>name</i>}</code>	Abfragezeichenfolgennamen unterscheiden zwischen Groß- und Kleinschreibung. API Gateway kombiniert mehrere Werte mit Kommas, z. B. "querystring1" "Value1,Value2" .
Anforderungstext	<code>\$request.body.<i>name</i></code> oder <code>\${request.body.<i>name</i>}</code>	Ein JSON-Pfadausdruck. Rekursiver Abstieg (<code>\$request.body..name</code>) und Filterausdrücke (<code>(?expression)</code>) werden nicht unterstützt.

 **Note**

Wenn Sie einen JSON-Pfad angeben, schneidet API Gateway den Anfragetext auf

Typ	Syntax	Hinweise
		100 KB ab und wendet dann den Auswahlausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an <code>\$request.body</code> .
Anforderungspfad	<code>\$request.path</code> oder <code>\${request.path}</code>	Der Anforderungspfad ohne den Namen der Stufe.
Path-Parameter	<code>\$request.path.name</code> oder <code>\${request.path.name}</code>	Der Wert eines Pfadparameters in der Anforderung. Beispiel: Wenn die Route <code>/pets/{petId}</code> ist, können Sie den Parameter <code>petId</code> aus der Anforderung mit <code>\$request.path.petId</code> zuordnen.
Kontextvariable	<code>\$context.variableName</code> oder <code>\${context.variableName}</code>	Der Wert einer Kontextvariablen .  Note NEs werden ausschließlich die Sonderzeichen <code>.</code> und <code>_</code> unterstützt.
Stufenvariable	<code>\$stageVariables.variableName</code> oder <code>\${stageVariables.variableName}</code>	Der Wert einer Stufenvariablen .
Statischer Wert	<code>string</code>	Ein konstanter Wert.

 Note

Um mehrere Variablen in einem Auswahlausdruck zu verwenden, schließen Sie die Variable in Klammern ein. Beispiel, `${request.path.name} ${request.path.id}`.

Transformieren von API-Antworten

Sie verwenden Antwortparameter, um die HTTP-Antwort einer Backend-Integration zu transformieren, bevor Sie die Antwort an Clients zurückgeben. Sie können Header oder den Statuscode einer Antwort ändern, bevor API Gateway die Antwort an Clients zurückgibt.

Sie konfigurieren die Antwortparameter für jeden Statuscode, den Ihre Integration zurückgibt. Antwortparameter sind eine Schlüssel-Wert-Zuordnung. Der Schlüssel gibt den Speicherort des zu ändernden Anforderungsparameters und an, wie er geändert werden soll. Der Wert gibt die neuen Daten für den Parameter an.

Die folgende Tabelle zeigt unterstützte Schlüssel.

Typ	Syntax
Header	<code>append overwrite remove:header. <i>headername</i></code>
Statuscode	<code>overwrite:statuscode</code>

Die folgende Tabelle zeigt unterstützte Werte, die Sie Parametern zuordnen können.

Typ	Syntax	Hinweise
Header-Wert	<code>\$response.header.<i>Name</i></code> oder <code>\${response.header.<i>name</i>}</code>	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. "header1" : "value1,value2" .

Typ	Syntax	Hinweise
		Einige Header sind reserviert. Weitere Informationen hierzu finden Sie unter the section called “Reservierte Header” .
Antworttext	<code>\$response.body.<i>name</i></code> oder <code>\${response.body.<i>name</i>}</code>	Ein JSON-Pfadausdruck. Rekursiver Abstieg (<code>\$response.body..name</code>) und Filterausdrücke (<code>(?expression)</code>) werden nicht unterstützt.  Note Wenn Sie einen JSON-Pfad angeben, kürzt API Gateway den Antworttext auf 100 KB und wendet dann den Auswahl Ausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an <code>\$response.body</code>.
Kontextvariable	<code>\$context.<i>variableName</i></code> oder <code>\${context.<i>variableName</i>}</code>	Der Wert einer unterstützten Kontextvariablen .
Stufenvariable	<code>\$stageVariables.<i>variableName</i></code> oder <code>\${stageVariables.<i>variableName</i>}</code>	Der Wert einer Stufenvariablen .
Statischer Wert	<i>string</i>	Ein konstanter Wert.

 Note

Um mehrere Variablen in einem Auswahlausdruck zu verwenden, schließen Sie die Variable in Klammern ein. Beispiel, `${request.path.name} ${request.path.id}`.

Reservierte Header

Die folgenden Header sind reserviert. Sie können keine Anforderungs- oder Antwort-Mappings für diese Header konfigurieren.

- access-control-*
- apigw-*
- Autorisierung
- Verbindung
- Content-Encoding
- Content-Length
- Content-Location
- Forwarded
- Keep-Alive
- Ursprung
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-*
- x-amzn-*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

Beispiele

Mit den folgenden AWS CLI-Beispielen werden Parameter-Mappings konfiguriert. CloudFormation-Beispielvorlagen finden Sie auf [GitHub](#).

Hinzufügen eines Headers zu einer API-Anforderung

Der folgende [create-integration](#)-Befehl erstellt einen Header mit dem Namen `header1` für eine API-Anfrage, bevor diese Ihre Backend-Integration erreicht. API Gateway füllt den Header mit der Anforderungs-ID auf.

```
aws apigatewayv2 create-integration \  
  --api-id abcdef123 \  
  --integration-type HTTP_PROXY \  
  --payload-format-version 1.0 \  
  --integration-uri 'https://api.example.com' \  
  --integration-method ANY \  
  --request-parameters '{ "append:header.header1": "$context.requestId" }'
```

Umbenennen eines Anforderungsheaders

Der folgende [create-integration](#)-Befehl benennt einen Anfrage-Header von `header1` in `header2` um:

```
aws apigatewayv2 create-integration \  
  --api-id abcdef123 \  
  --integration-type HTTP_PROXY \  
  --payload-format-version 1.0 \  
  --integration-uri 'https://api.example.com' \  
  --integration-method ANY \  
  --request-parameters '{ "append:header.header2": "$request.header.header1",  
    "remove:header.header1": "" }'
```

Ändern der Antwort aus einer Integration

Der folgende [create-integration](#)-Befehl konfiguriert Antwortparameter für eine Integration. Wenn die Integrationen einen 500-Statuscode zurückgeben, ändert API Gateway den Statuscode auf 403 und fügt der Antwort `header11` hinzu. Wenn die Integration einen 404-Statuscode zurückgibt, fügt API Gateway der Antwort einen `error`-Header hinzu.

```
aws apigatewayv2 create-integration \  
  --api-id abcdef123 \  
  --response-parameters '{ "response:500:header.header11": "$response.body",  
    "response:404:error": "$response.body" }'
```

```
--integration-type HTTP_PROXY \  
--payload-format-version 1.0 \  
--integration-uri 'https://api.example.com' \  
--integration-method ANY \  
--response-parameters '{"500" : {"append:header.header1": "$context.requestId",  
"overwrite:statusCode" : "403"}, "404" : {"append:header.error" :  
"$stageVariables.environmentId"} }'
```

Entfernen konfigurierter Parameter-Mappings

Der folgende [update-integration](#)-Befehl entfernt zuvor konfigurierte Anforderungsparameter für `append:header.header1`. Außerdem werden zuvor konfigurierte Antwortparameter für einen 200-Statuscode entfernt.

```
aws apigatewayv2 update-integration \  
--api-id abcdef123 \  
--integration-id hijk456 \  
--request-parameters '{"append:header.header1" : ""}' \  
--response-parameters '{"200" : {}}'
```

Verwenden Sie OpenAPI-Definitionen für HTTP APIs in API Gateway

Sie können Ihre HTTP-API definieren, indem Sie eine OpenAPI 3.0-Definitionsdatei verwenden. Anschließend können Sie die Definition in API Gateway importieren, um eine API zu erstellen. Weitere Informationen zu API Gateway-Erweiterungen für OpenAPI finden Sie unter [Erweiterungen für OpenAPI](#).

Importieren einer HTTP-API

Sie können eine HTTP-API erstellen, indem Sie eine OpenAPI 3.0-Definitionsdatei importieren.

Um von einer REST-API zu einer HTTP-API zu migrieren, können Sie Ihre REST-API als OpenAPI 3.0-Definitionsdatei exportieren. Importieren Sie dann die API-Definition als HTTP-API. Weitere Informationen zum Exportieren einer REST-API finden Sie unter [REST-API von API Gateway importieren](#).

Note

HTTP APIs unterstützt dieselben AWS Variablen wie REST APIs. Weitere Informationen hierzu finden Sie unter [AWS Variablen für OpenAPI-Import](#).

Importieren von Validierungsinformationen

Beim Importieren einer API stellt API Gateway drei Kategorien von Validierungsinformationen bereit.

Info

Eine Eigenschaft ist gemäß der OpenAPI-Spezifikation gültig, aber diese Eigenschaft wird für HTTP APIs nicht unterstützt.

Das folgende OpenAPI 3.0-Snippet erzeugt beispielsweise Informationen beim Import, da HTTP die Überprüfung von Anfragen APIs nicht unterstützt. API Gateway ignoriert die Felder `requestBody` und `schema`.

```
"paths": {
  "/": {
    "get": {
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
        "payloadFormatVersion": "1.0"
      },
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Body"
            }
          }
        }
      }
    }
  },
  ...
},
"components": {
  "schemas": {
    "Body": {
      "type": "object",
      "properties": {
        "key": {
          "type": "string"
        }
      }
    }
  }
}
```



```
    }  
  }  
  ...  
}  
...  
}
```

Warnung

Eine Eigenschaft oder Struktur ist gemäß der OpenAPI-Spezifikation ungültig, blockiert die API-Erstellung jedoch nicht. Sie können angeben, ob API Gateway diese Warnungen ignorieren und mit der API-Erstellung fortfahren oder die API-Erstellung bei Warnungen beenden soll.

Das folgende OpenAPI 3.0-Dokument erzeugt Warnungen beim Import, da HTTP nur Lambda-Proxy- und HTTP-Proxyintegrationen APIs unterstützt.

```
"x-amazon-apigateway-integration": {  
  "type": "AWS",  
  "httpMethod": "POST",  
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",  
  "payloadFormatVersion": "1.0"  
}
```

Fehler

Die OpenAPI-Spezifikation ist ungültig oder fehlerhaft. API Gateway kann keine Ressourcen aus dem fehlerhaften Dokument erstellen. Sie müssen die Fehler beheben und es anschließend erneut versuchen.

Die folgende API-Definition erzeugt Fehler beim Import, da HTTP nur die OpenAPI 3.0-Spezifikation APIs unterstützt.

```
{  
  "swagger": "2.0.0",  
  "info": {  
    "title": "My API",  
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",  
    "version": "1.0"  
  }  
  ...  
}
```

Ein weiteres Beispiel: Während OpenAPI es Benutzern ermöglicht, eine API mit mehreren Sicherheitsanforderungen zu definieren, die mit einer bestimmten Operation verbunden sind, wird dies von API Gateway nicht unterstützt. Jede Operation kann nur eine IAM-Autorisierung, einen Lambda-Genehmiger oder einen JWT-Genehmiger haben. Der Versuch, mehrere Sicherheitsanforderungen zu modellieren, führt zu einem Fehler.

Importieren Sie eine API mithilfe der AWS CLI

Der folgende [import-api](#)-Befehl importiert die OpenAPI 3.0-Definitionsdatei `api-definition.json` als HTTP-API:

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

Sie können das folgende Beispiel einer OpenAPI 3.0-Definition importieren, um eine HTTP-API zu erstellen.

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",

```

```
        "schema": {
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Pets"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "HTTP_PROXY",
        "httpMethod": "GET",
        "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
        "payloadFormatVersion": 1.0
      }
    },
    "post": {
      "operationId": "Create Pet",
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/NewPet"
            }
          }
        }
      },
      "required": true
    },
    "responses": {
```

```

    "200": {
      "description": "200 response",
      "headers": {
        "Access-Control-Allow-Origin": {
          "schema": {
            "type": "string"
          }
        }
      },
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/NewPetResponse"
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "type": "HTTP_PROXY",
      "httpMethod": "POST",
      "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
      "payloadFormatVersion": 1.0
    }
  },
  "/pets/{petId}": {
    "get": {
      "operationId": "Get Pet",
      "parameters": [
        {
          "name": "petId",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {

```

```

        "schema": {
            "type": "string"
        }
    },
    "content": {
        "application/json": {
            "schema": {
                "$ref": "#/components/schemas/Pet"
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "type": "HTTP_PROXY",
    "httpMethod": "GET",
    "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/{petId}",
    "payloadFormatVersion": 1.0
}
},
"x-amazon-apigateway-cors": {
    "allowOrigins": [
        "*"
    ],
    "allowMethods": [
        "GET",
        "OPTIONS",
        "POST"
    ],
    "allowHeaders": [
        "x-amzm-header",
        "x-apigateway-header",
        "x-api-key",
        "authorization",
        "x-amz-date",
        "content-type"
    ]
},
"components": {
    "schemas": {

```

```
"Pets": {
  "type": "array",
  "items": {
    "$ref": "#/components/schemas/Pet"
  }
},
"Empty": {
  "type": "object"
},
"NewPetResponse": {
  "type": "object",
  "properties": {
    "pet": {
      "$ref": "#/components/schemas/Pet"
    },
    "message": {
      "type": "string"
    }
  }
},
"Pet": {
  "type": "object",
  "properties": {
    "id": {
      "type": "string"
    },
    "type": {
      "type": "string"
    },
    "price": {
      "type": "number"
    }
  }
},
"NewPet": {
  "type": "object",
  "properties": {
    "type": {
      "$ref": "#/components/schemas/PetType"
    },
    "price": {
      "type": "number"
    }
  }
}
```

```

    },
    "PetType": {
      "type": "string",
      "enum": [
        "dog",
        "cat",
        "fish",
        "bird",
        "gecko"
      ]
    }
  }
}
}
}

```

Exportieren Sie HTTP APIs aus dem API Gateway

Nachdem Sie ein HTTP-API erstellt haben, können Sie eine OpenAPI 3.0-Definition Ihrer API aus API Gateway exportieren. Sie können entweder eine Stufe auswählen, die exportiert werden soll, oder die neueste Konfiguration Ihrer API exportieren. Sie können auch eine exportierte API-Definition nach API Gateway importieren, um eine weitere, identische API zu erstellen. Weitere Informationen zum Importieren von API-Definitionen finden Sie unter [Importieren einer HTTP-API](#).

Exportieren Sie eine OpenAPI 3.0-Definition einer Phase mithilfe der CLI AWS

Der folgende [export-api](#)-Befehl exportiert eine OpenAPI-Definition einer API-Stufe namens `prod` in eine YAML-Datei mit dem Namen `stage-definition.yaml`. Die exportierte Definitionsdatei enthält standardmäßig [API Gateway-Erweiterungen](#).

```

aws apigatewayv2 export-api \
  --api-id api-id \
  --output-type YAML \
  --specification OAS30 \
  --stage-name prod \
  stage-definition.yaml

```

Exportieren Sie eine OpenAPI 3.0-Definition der neuesten Änderungen Ihrer API mithilfe der CLI AWS

Der folgende [export-api](#)-Befehl exportiert eine OpenAPI-Definition einer HTTP-API in eine JSON-Datei mit dem Namen `latest-api-definition.json`. Da der Befehl keine Stufe angibt,

exportiert API Gateway die neueste Konfiguration Ihrer API, unabhängig davon, ob sie auf einer Stufe bereitgestellt wurde oder nicht. Die exportierte Definitionsdatei enthält keine [API Gateway-Erweiterungen](#).

```
aws apigatewayv2 export-api \  
  --api-id api-id \  
  --output-type JSON \  
  --specification OAS30 \  
  --no-include-extensions \  
  latest-api-definition.json
```

Weitere Informationen finden Sie unter [ExportAPI](#) in der API-Referenz für Amazon API Gateway Version 2.

Exportieren einer OpenAPI 3.0-Definition mithilfe der API-Gateway-Konsole

Das folgende Verfahren zeigt, wie Sie eine OpenAPI-Definition einer HTTP-API exportieren.

Exportieren einer OpenAPI 3.0-Definition mithilfe der API-Gateway-Konsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway>.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie im Hauptnavigationsbereich unter Entwickeln die Option Exportieren aus.
4. Wählen Sie aus den folgenden Optionen, um Ihre API zu exportieren:

[API Gateway](#) > [APIs](#) > [my-http-api \(abcdef1234\)](#) > **Export**

Export

Export an OpenAPI 3 definition [Info](#)

Download an OpenAPI 3 definition of your latest changes or a stage's configuration.

Source

Extensions [Learn more](#) [↗](#)

☒ Include API Gateway extensions

Output format

☒ JSON

☐ YAML

Download

- a. Wählen Sie unter Quelle eine Quelle für die OpenAPI 3.0-Definition aus. Sie können entweder eine Stufe auswählen, die exportiert werden soll, oder die neueste Konfiguration Ihrer API exportieren.
 - b. Aktivieren Sie API Gateway-Erweiterungen einschließen, um [API-Gateway-Erweiterungen](#) einzubeziehen.
 - c. Wählen Sie unter Ausgabeformat ein Ausgabeformat aus.
5. Wählen Sie Herunterladen aus.

HTTP-APIs für Kundenaufrufe veröffentlichen

Sie können Stufen und benutzerdefinierte Domännennamen verwenden, um Ihre API zu veröffentlichen, die Clients aufrufen können.

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. dev, prod, beta oder v2). Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt. Sie können verschiedene Integrationen und Einstellungen für jede Stufe einer API konfigurieren.

Sie können benutzerdefinierte Domännennamen verwenden, um eine einfachere, intuitivere URL für Clients bereitzustellen, die Ihre API aufrufen können, als die Standard-UR, `https://api-id.execute-api.region.amazonaws.com/stage`.

Note

Um die Sicherheit Ihrer API-Gateway-APIs zu erhöhen, ist die `execute-api.{region}.amazonaws.com`-Domain in der [Public Suffix List \(PSL\)](#) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host--`-Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domain-Namen für Ihre API-Gateway-APIs einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der [Set-Cookie](#)-Seite im Mozilla Developer Network.

Themen

- [Stufen für HTTP APIs im API Gateway](#)
- [Sicherheitsrichtlinie für HTTP APIs in API Gateway](#)
- [Benutzerdefinierte Domainnamen für HTTP-APIs in API Gateway](#)

Stufen für HTTP APIs im API Gateway

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. `dev`, `prod`, `beta` oder `v2`). API-Stufen werden durch ihre API-ID und den Stufennamen identifiziert und sind in der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

Sie können eine `$default`-Phase erstellen, die beispielsweise von der Basis der URL Ihrer API bereitgestellt wird, z. B. `https://{api_id}.execute-api.{region}.amazonaws.com/`. Sie verwenden diese URL, um eine API-Phase aufzurufen.

Eine Bereitstellung ist ein Snapshot Ihrer API-Konfiguration. Nachdem Sie eine API in einer Phase bereitgestellt haben, kann sie von Clients aufgerufen werden. Sie müssen eine API bereitstellen, damit Änderungen wirksam werden. Wenn Sie automatische Bereitstellungen aktivieren, werden Änderungen an einer API automatisch für Sie freigegeben.

Verwenden Sie Stage-Variablen für HTTP APIs in API Gateway

Stufenvariablen sind Schlüssel-Werte-Paare, die Sie für eine Stufe einer HTTP-API definieren können. Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung verwendet werden.

Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Genehmigers übergeben. Weitere Informationen hierzu finden Sie unter [the section called “Antwortformat des Lambda-Genehmigers”](#).

Beispiel – Den HTTP-Integrationsendpunkt mithilfe einer Stufenvariablen anpassen

Sie können beispielsweise eine Stufenvariable definieren und dann ihren Wert als HTTP-Endpunkt für eine HTTP-Proxy-Integration festlegen. Später können Sie den Endpunkt mithilfe des zugeordneten Stufenvariablennamens referenzieren. Auf diese Weise können Sie in jeder Stufe dieselbe API-Einrichtung mit einem anderen Endpunkt verwenden. In ähnlicher Weise können Sie Stufenvariablen verwenden, um für jede Phase Ihrer API eine andere AWS Lambda Funktionsintegration anzugeben.

Wenn Sie eine Stufenvariable zum Anpassen des HTTP-Integrationsendpunkts verwenden möchten, müssen Sie zunächst den Namen und den Wert der Stufenvariablen (z. B. `url`) mit dem Wert `example.com` festlegen. Richten Sie als Nächstes eine HTTP-Proxy-Integration ein. Anstatt die URL des Endpunkts einzugeben, können Sie API Gateway anweisen, den Wert der Stufenvariablen zu verwenden, **`http://${stageVariables.url}`**. Dieser Wert weist API Gateway an, Ihre Stufenvariable `${}` zur Laufzeit abhängig von der Stufe Ihrer API zu ersetzen.

Sie können auf ähnliche Weise auf Stufenvariablen verweisen, um einen Lambda-Funktionsnamen oder einen AWS Rollen-ARN anzugeben.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Der folgende [add-permission](#)-Befehl konfiguriert die Berechtigung für die Lambda-Funktion:

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action lambda:InvokeFunction
```

API Gateway-Stufenvariablenreferenz für HTTP APIs in API Gateway

In den folgenden Fällen können Sie API Gateway APIs Gateway-Stufenvariablen für HTTP verwenden.

HTTP-Integration URIs

Sie können eine Stufenvariable als Teil einer HTTP-Integrations-URI verwenden, wie in den folgenden Beispielen gezeigt.

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Lambda-Funktionen

Sie können eine Stufenvariable anstelle eines Integrationsnamens oder Alias für die Lambda-Funktion verwenden, wie in den folgenden Beispielen gezeigt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

Note

Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

AWS Anmeldeinformationen für die Integration

Sie können eine Stage-Variable als Teil eines ARN mit AWS Benutzer- oder Rollenmeldedaten verwenden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Sicherheitsrichtlinie für HTTP APIs in API Gateway

API Gateway erzwingt eine Sicherheitsrichtlinie von TLS_1_2 für alle HTTP-API-Endpunkte.

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus einer TLS-Mindestversion und Verschlüsselungsverfahren, die von Amazon API Gateway bereitgestellt wird. Das TLS-Protokoll behandelt Netzwerksicherheitsprobleme wie Manipulationen und Abhören zwischen einem Client und einem Server. Wenn Ihre Clients über die benutzerdefinierte Domäne einen TLS-Handshake mit Ihrer API ausführen, erzwingt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite-Optionen, die von Ihren Clients verwendet werden können. Diese Sicherheitsrichtlinie akzeptiert TLS 1.2- und TLS 1.3-Verkehr und weist TLS 1.0-Verkehr zurück.

Unterstützte TLS-Protokolle und Chiffren für HTTP APIs

In der folgenden Tabelle werden die unterstützten TLS-Protokolle für HTTP beschrieben. APIs

TLS-Protokolle	TLS_1_2-Sicherheit srichtlinie
TLSv13.	Ja
TLSv12.	Ja

In der folgenden Tabelle werden die TLS-Verschlüsselungen beschrieben, die für die Sicherheitsrichtlinie TLS 1_2 für HTTP verfügbar sind. APIs

TLS-Verschlüsselungsverfahren	TLS_1_2-Sicherheit srichtlinie
TLS-AES-128-GCM- SHA256	Ja

TLS-Verschlüsselungsverfahren	TLS_1_2-Sicherheitsrichtlinie
TLS-AES-256-GCM- SHA384	Ja
TLS- - - CHACHA20 POLY1305 SHA256	Ja
ECDHE-ECDSA- -GCM- AES128 SHA256	Ja
ECDHE-RSA- AES128 -GCM- SHA256	Ja
ECDHE-ECDSA- AES128 - SHA256	Ja
ECDHE-RSA- - AES128 SHA256	Ja
ECDHE-ECDSA- -GCM AES256 - SHA384	Ja
ECDHE-RSA- AES256 -GCM- SHA384	Ja
ECDHE-ECDSA- AES256 - SHA384	Ja
ECDHE-RSA- - AES256 SHA384	Ja
AES128-GCM- SHA256	Ja
AES128-SHA256	Ja
AES256-GCM- SHA384	Ja
AES256-SHA256	Ja

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für die gleichen Verschlüsselungsverfahren. Eine Liste der Namen der Verschlüsselungsverfahren finden Sie unter [the section called “OpenSSL- und RFC-Verschlüsselungsnamen”](#).

Informationen über REST und APIs WebSocket APIs

Weitere Informationen zu REST APIs und WebSocket APIs finden Sie unter [the section called “Eine Sicherheitsrichtlinie auswählen”](#) und [the section called “Sicherheitsrichtlinie für WebSocket APIs in API Gateway”](#).

Benutzerdefinierte Domainnamen für HTTP-APIs in API Gateway

Benutzerdefinierte Domainnamen sind einfachere und intuitivere URLs, die Sie Ihren API-Benutzern zur Verfügung stellen können.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

hierzu wird *api-id* in API Gateway generiert, *region* ist die AWS-Region und *stage* wird als Stufe angegeben wenn Sie die API bereitstellen.

Der Hostname-Teil der URL (also *api-id*.execute-api.*region*.amazonaws.com) verweist auf einen API-Endpunkt. Der Standard-API-Endpunkt hat möglicherweise einen schwer zu merkenden, zufallsgenerierten Namen, der nicht besonders benutzerfreundlich ist.

Mit benutzerdefinierten Domainnamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. myservice) auswählen, um die alternative URL Ihrer API zuzuordnen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

Überlegungen

Die folgenden Überlegungen können sich auf Ihre Verwendung eines benutzerdefinierten Domainnamens auswirken.

- Eine regionale benutzerdefinierte Domain kann REST-APIs and HTTP-APIs zugeordnet werden. Sie können API-Gateway-Version-2-APIs verwenden, um regionale benutzerdefinierte Domainnamen für REST-APIs zu erstellen und zu verwalten.
- Als Mindest-TLS-Version wird ausschließlich TLS 1.2 unterstützt.

- Sie müssen den Ressourceneintrag Ihres DNS-Anbieters erstellen oder aktualisieren, bevor Sie ihn dem API-Endpunkt zuordnen können. Ohne ein solches Mapping können API-Anfragen für den benutzerdefinierten Domainnamen API Gateway nicht erreichen.
- Mit einem Wildcard-Zertifikat können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das Standardkontingent zu überschreiten. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen mit Platzhalter”](#).

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Erstellung eines benutzerdefinierten Domainnamens aufgeführt.

Registrieren eines Domainnamens

Wenn Sie benutzerdefinierte Domainnamen für Ihre APIs einrichten möchten, müssen Sie eine Internetdomäne registrieren. Sie können Ihre Internetdomain mithilfe von [Amazon Route 53](#) oder mit einem Domain-Registrar Ihrer Wahl registrieren. Ihr benutzerdefinierter Domainname kann der Name einer Unter- oder Root-Domain (Zone Apex) einer registrierten Internetdomain sein.

Ihr Domainname muss der [RFC 1035](#)-Spezifikation entsprechen und darf ein Maximum von 63 Oktetten pro Label und insgesamt 255 Oktetten nicht überschreiten.

Zertifikate für benutzerdefinierte Domainnamen

Vor der Einrichtung eines benutzerdefinierten Domainnamens für eine API müssen Sie ein SSL-/TLS-Zertifikat in ACM vorbereiten. Wenn ACM in der AWS-Region nicht verfügbar ist, in der Sie Ihren benutzerdefinierten Domainnamen erstellen, müssen Sie ein Zertifikat für API Gateway in dieser Region importieren.

Wenn Sie ein SSL-/TLS-Zertifikat importieren möchten, müssen Sie den PEM-formatierten Hauptteil des SSL-/TLS-Zertifikats, den persönlichen Schlüssel und die Zertifikatkette für den benutzerdefinierten Domainnamen bereitstellen.

Ein in ACM gespeichertes Zertifikat wird durch seinen ARN angegeben. Wenn Sie Zertifikate verwenden, die von ACM ausgestellt werden, brauchen sich keine Gedanken über vertrauliche Zertifikatdetails wie private Schlüssel zu machen. Wenn Sie also ein von AWS verwaltetes Zertifikat für einen Domainnamen verwenden möchten, müssen Sie lediglich auf dessen ARN verweisen.

Wenn Ihre Anwendung, manchmal auch als Zertifikats-Pinning bekannt, verwendet, um ein ACM-Zertifikat zu pinnen, kann die Anwendung möglicherweise keine Verbindung zu Ihrer Domain

herstellen, nachdem AWS das Zertifikat erneuert hat. Weitere Informationen finden Sie unter [Probleme beim Zertifikats-Pinning](#) im AWS Certificate Manager-Benutzerhandbuch.

Benutzerdefinierte Domainnamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domainnamen können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das [Standardkontingent](#) zu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domainnamen geben, *customername*.api.example.com.

Um einen benutzerdefinierten Platzhalterdomainnamen zu erstellen, geben Sie einen Platzhalter (*) als erste Subdomain einer benutzerdefinierten Domain an, die alle möglichen Subdomains einer Root-Domain darstellt.

Der benutzerdefinierte Domainnamen mit Platzhalter *.example.com führt beispielsweise zu Unterdomains wie a.example.com, b.example.com und c.example.com, die alle zur gleichen Domain weiterleiten.

Benutzerdefinierte Domainnamen mit Platzhaltern unterstützen andere Konfigurationen als die benutzerdefinierten Standarddomainnamen von API Gateway. Beispielsweise können Sie in einem einzelnen AWS-Konto *.example.com und a.example.com so konfigurieren, dass sie sich anders verhalten.

Um einen benutzerdefinierten Domänennamen mit Platzhalter zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat angeben, das mithilfe der DNS- oder der E-Mail-Validierungsmethode validiert wurde.

Note

Sie können keinen benutzerdefinierten Domainnamen mit Platzhalter erstellen, wenn ein anderes AWS-Konto einen benutzerdefinierten Domainnamen erstellt hat, der mit dem benutzerdefinierten Domainnamen mit Platzhalter in Konflikt steht. Wurde a.example.com beispielsweise von Konto A erstellt, dann kann Konto B als benutzerdefinierten Domainnamen mit Platzhalter nicht *.example.com erstellen.

Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das [AWS Supportcenter](#) wenden, um eine Ausnahme anzufordern.

Die nächsten Schritte für benutzerdefinierte Domainnamen

Verwenden Sie die Dokumentation aus dem REST-API-Abschnitt des API-Gateway-Entwicklerhandbuchs für die Einrichtung eines benutzerdefinierten Domainnamens für eine HTTP-API.

Geben Sie zunächst ein Zertifikat für Ihren benutzerdefinierten Domainnamen an. Weitere Informationen finden Sie unter [the section called “Bereiten Sie Zertifikate vor in AWS Certificate Manager”](#). Dann erstellen Sie einen regionalen benutzerdefinierten Domainnamen. Weitere Informationen finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

API-Stufen zu einem benutzerdefinierten Domainnamen für HTTP-APIs zuweisen

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Nachdem Sie einen Domain-Namen erstellt und DNS-Einträge konfiguriert haben, verwenden Sie API-Mappings, um Datenverkehr über Ihren benutzerdefinierten Domain-Namen an Ihre APIs zu senden.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise die production-Phase einer API in `https://api.example.com/orders` abbilden.

Sie können HTTP-API- und REST-API--Stufen demselben benutzerdefinierten Domain-Namen zuweisen.

Bevor Sie ein API-Mapping erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

Weiterleiten von API-Anforderungen

Sie können API-Mappings mit mehreren Ebenen konfigurieren, z. B. `orders/v1/items` und `orders/v2/items`.

Bei API-Mappings mit mehreren Ebenen leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspfad hat. API Gateway berücksichtigt nur die für API-Mappings konfigurierten Pfade und keine API-Routen, um die aufzurufende API auszuwählen. Wenn kein Pfad mit der Anforderung übereinstimmt, sendet API Gateway die Anforderung an die API, die Sie dem leeren Pfad (none) zugeordnet haben.

Bei benutzerdefinierten Domain-Namen, die API-Mappings mit mehreren Ebenen verwenden, leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspräfix hat.

Betrachten Sie beispielsweise einen benutzerdefinierten Domain-Namen `https://api.example.com` mit den folgenden API-Mappings:

1. (none) API 1 zugewiesen.
2. `orders` API 2 zugewiesen.
3. `orders/v1/items` API 3 zugewiesen.
4. `orders/v2/items` API 4 zugewiesen.
5. `orders/v2/items/categories` API 5 zugewiesen.

Anfrage	Ausgewählte API	Erklärung
<code>https://api.example.com/orders</code>	API 2	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v1/items</code>	API 3	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v2/items</code>	API 4	Die Anforderung stimmt genau mit diesem API-Mapping überein.
<code>https://api.example.com/orders/v1/items/123</code>	API 3	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat. Das 123 am Ende der Anforderung hat keinen Einfluss auf die Auswahl.
<code>https://api.example.com/orders/v2/items/categories/5</code>	API 5	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat.

Anfrage	Ausgewählte API	Erklärung
<code>https://api.example.com/customers</code>	API 1	API Gateway verwendet die leere Zuweisung als Catch-All.
<code>https://api.example.com/ordersandmore</code>	API 2	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspräfix hat. Bei einem benutzerdefinierten Domain-Namen, der mit einstufigen Mappings konfiguriert ist, z. B. nur <code>https://api.example.com/orders</code> und <code>https://api.example.com/</code>, würde API Gateway API 1 auswählen, da es keinen passenden Pfad mit <code>ordersandmore</code> gibt.

Einschränkungen

- In einer API-Zuweisung müssen sich der benutzerdefinierte Domainname und die zugeordneten APIs im selben AWS-Konto befinden.
- API-Zuweisungen dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: `$-_.+!*'()/`.
- Die maximale Länge für den Pfad in einer API-Zuweisung beträgt 300 Zeichen.
- Es können 200 API-Zuweisungen mit mehreren Ebenen für jeden Domainnamen vorhanden sein. Dieses Limit beinhaltet keine API-Zuweisungen mit einer einzigen Ebene, wie z. B. `/prod`.
- Sie können HTTP-APIs nur einem regionalen benutzerdefinierten Domainnamen mit der TLS 1.2-Sicherheitsrichtlinie zuordnen.
- Sie können WebSocket-APIs nicht demselben benutzerdefinierten Domainnamen wie dem einer HTTP-API oder REST-API zuweisen.

- Wenn Sie mehrstufige API-Zuweisungen erstellen, konvertiert API Gateway alle Header-Namen in Kleinbuchstaben.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

AWS Serverless Application Model-Beispielvorlagen, die alle Ressourcen erstellen, finden Sie unter [Sitzungen mit SAM](#) auf GitHub.

AWS-Managementkonsole

So erstellen Sie ein API-Mapping

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie benutzerdefinierte Domain-Namen aus.
3. Wählen Sie einen benutzerdefinierten Domain-Namen aus, den Sie bereits erstellt haben.
4. Wählen Sie API-Mappings aus.
5. Wählen Sie API-Zuweisungen konfigurieren aus.
6. Wählen Sie Neue Zuweisung hinzufügen aus.
7. Geben Sie eine API, eine Phase und optional einen Pfad ein.
8. Wählen Sie Speichern aus.

AWS CLI

Der folgende [create-api-mapping](#)-Befehl erstellt eine API-Zuweisung. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1/orders` an die angegebene API und Phase.

```
aws apigatewayv2 create-api-mapping \  
  --domain-name api.example.com \  
  --api-mapping-key v1/orders \  
  --api-id a1b2c3d4 \  
  --phase-name default
```

```
--stage test
```

CloudFormation

Mit dem folgenden CloudFormation-Beispiel wird ein API-Mapping erstellt.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Deaktivieren des Standardendpunkts für REST-APIs

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domänennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt. Wenn Sie den Standardendpunkt deaktivieren, wirkt sich dies auf alle Stufen einer API aus.

Der folgende Vorgang zeigt, wie ein Standardendpunkt für eine HTTP-API deaktiviert wird.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie die ID Ihrer API aus, um die Seite mit den API-Details zu öffnen.
4. Klicken Sie unter API-Details auf Bearbeiten.
5. Klicken Sie unter Standardendpunkt auf Deaktivieren.
6. Wählen Sie Speichern aus.

Wenn Sie für Ihre Stufe automatische Bereitstellungen aktivieren, müssen Sie Ihre API nicht erneut bereitstellen, damit die Änderung wirksam wird. Andernfalls müssen Sie Ihre API erneut bereitstellen.

7. (Optional) Klicken Sie auf Bereitstellen und stellen Sie Ihre API erneut bereit oder erstellen Sie eine neue Stufe, in der die Änderung wirksam werden soll.

AWS CLI

Der [update-domain-name](#)-Befehl deaktiviert den Standardendpunkt einer HTTP-API:

```
aws apigatewayv2 update-api \  
  --api-id abcdef123 \  
  --disable-execute-api-endpoint
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird, es sei denn, automatische Bereitstellungen sind aktiviert.

Der Befehl [create-deployment](#) erstellt eine Bereitstellung:

```
aws apigatewayv2 create-deployment \  
  --api-id abcdef123 \  
  --stage-name dev
```

IP-Adresstypen für benutzerdefinierte Domainnamen für HTTP-APIs

Beim Erstellen einer API geben Sie den Typ der IP-Adressen an, die Ihre Domain aufrufen können. Sie können IPv4 auswählen, um IPv4-Adressen für den Aufruf Ihrer Domain zuzulassen, oder Dualstack, um sowohl IPv4- als auch IPv6-Adressen für den Aufruf zuzulassen. Wir empfehlen, den IP-Adresstyp auf Dualstack zu setzen, um Engpässe im IP-Adressraum zu vermeiden und Ihre Sicherheitsposition zu verbessern. Weitere Informationen zu den Vorteilen eines Dualstack-IP-Adresstyps finden Sie unter [IPv6 in AWS](#).

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können Ihre Verwendung von IP-Adresstypen beeinflussen.

- Der Standard-IP-Adresstyp für benutzerdefinierte Domainnamen in API Gateway ist IPv4.
- Ihr benutzerdefinierter Domainname muss nicht denselben IP-Adresstyp für alle darauf zugewiesenen APIs haben. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.

Ändern des IP-Adresstyps eines benutzerdefinierten Domainnamens

Sie können den IP-Adresstyp ändern, indem Sie die Endpunktkonfiguration der Domain aktualisieren. Dies kann über die AWS-Managementkonsole, die AWS CLI, CloudFormation oder ein AWS SDK erfolgen.

AWS-Managementkonsole

So ändern Sie den IP-Adresstyp eines benutzerdefinierten Domainnamens

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie einen öffentlichen benutzerdefinierten Domainnamen aus.
3. Wählen Sie Endpunktkonfiguration aus.
4. Wählen Sie unter „IP-Adresstyp“ entweder IPv4 oder Dualstack aus.
5. Wählen Sie Speichern aus.

AWS CLI

Der folgende [update-domain-name](#)-Befehl aktualisiert eine API, sodass sie den IP-Adresstyp „Dualstack“ aufweist:

```
aws apigatewayv2 update-domain-name \  
  --domain-name dualstack.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "ApiMappingSelectionExpression": "$request.basepath",  
  "DomainName": "dualstack.example.com",  
  "DomainNameConfigurations": [  
    {  
      "ApiGatewayDomainName": "d-abcd1234.execute-api.us-  
east-1.amazonaws.com",  
      "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/  
abcd1234-5678-abc",  
      "DomainNameStatus": "AVAILABLE",  
      "EndpointType": "REGIONAL",
```



```
        "HostedZoneId": "Z3LQWSYCGH4ADY",
        "SecurityPolicy": "TLS_1_2",
        "IpAddressType": "dualstack"
    }
],
"Tags": {}
}
```

Schutz Ihrer HTTP-APIs in API Gateway

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, wie böswillige Benutzer oder Spitzen im Datenverkehr. Sie können Ihre API mithilfe von Strategien wie das Festlegen von Drosselungs-Targets und das Aktivieren von gegenseitiger TLS schützen. In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen mit API Gateway aktivieren können.

Themen

- [Drosseln Sie Anfragen an Ihr HTTP APIs für einen besseren Durchsatz in API Gateway](#)
- [So aktivieren Sie die gegenseitige TLS-Authentifizierung für Ihr HTTP APIs in API Gateway](#)

Drosseln Sie Anfragen an Ihr HTTP APIs für einen besseren Durchsatz in API Gateway

Sie können die Drosselung für Sie konfigurieren APIs , um sie davor zu schützen, von zu vielen Anfragen überfordert zu werden. Drosselungen werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere untersucht API Gateway die Rate und die Anzahl der eingereichten Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise 429 Too Many Requests Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Zielgrenzwerte für einzelne API-Phasen oder -Routen festlegen, um die Gesamtleistung APIs in Ihrem Konto zu verbessern.

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) APIs innerhalb eines AWS Kontos pro Region. Außerdem wird der Burst (d. h. die maximale Bucket-Größe) für alle APIs innerhalb eines AWS Kontos pro Region begrenzt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von 429 Too Many Requests-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter [Kontingente und wichtige Hinweise](#).

Limits pro Konto gelten für alle APIs Konten in einer bestimmten Region. Das Ratenlimit auf Kontoebene kann auf Anfrage erhöht werden. Höhere Limits sind möglich, was kürzere Timeouts und geringere Payloads zur Folge hat. APIs Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das [AWS Supportcenter](#). Weitere Informationen finden Sie unter [Kontingente und wichtige Hinweise](#). Beachten Sie, dass diese Limits nicht höher als die Drosselungsgrenzen sein können. AWS

Drosselung auf Routenebene

Sie können die Drosselung auf Routenebene festlegen, um die Anforderungs-Drosselungslimits auf Kontoebene für eine bestimmte Stufe oder für individuelle Routen in Ihrer API zu überschreiben. Die Standardgrenzen für die Routendrosselung können die Ratenlimits auf Kontoebene nicht überschreiten.

Sie können die Drosselung auf Routenebene konfigurieren, indem Sie die verwerde AWS CLI. Der folgende [update-stage](#)-Befehl konfiguriert die benutzerdefinierte Drosselung für die angegebene Stufe und Route einer API.

```
aws apigatewayv2 update-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-settings '{"GET /pets":  
{"ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000}}'
```

So aktivieren Sie die gegenseitige TLS-Authentifizierung für Ihr HTTP APIs in API Gateway

Die gegenseitige TLS-Authentifizierung erfordert eine bidirektionale Authentifizierung zwischen dem Client und dem Server. Bei gegenseitiger TLS müssen Clients X.509-Zertifikate zur Prüfung ihrer Identität präsentieren, um auf Ihre API zuzugreifen. Gegenseitiges TLS ist eine allgemeine Anforderung für das Internet der Dinge (IoT) und business-to-business Anwendungen.

Sie können gegenseitiges TLS zusammen mit anderen [Autorisierungs- und Authentifizierungsvorgängen](#) verwenden, die API Gateway unterstützt. API Gateway leitet die von Kunden bereitgestellten Zertifikate an Lambda-Genehmiger und an Backend-Integrationen weiter.

Important

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Clients nur mithilfe eines benutzerdefinierten Domännennamens mit gegenseitiger TLS auf Ihre API zugreifen können, deaktivieren Sie den `execute-api`-Standardendpunkt. Weitere Informationen hierzu finden Sie unter [the section called “Deaktivieren des Standardendpunkts”](#).

Voraussetzungen für gegenseitige TLS

Um gegenseitige TLS zu konfigurieren, benötigen Sie:

- Einen benutzerdefinierten Domännennamen
- AWS Certificate Manager Für Ihren benutzerdefinierten Domainnamen ist mindestens ein Zertifikat konfiguriert
- Ein in Amazon S3 konfiguierter und hochgeladener Truststore

Benutzerdefinierte Domainnamen

Um die gegenseitige TLS-Authentifizierung für eine HTTP-API zu aktivieren, müssen Sie einen benutzerdefinierten Domännennamen für Ihre API konfigurieren. Sie können gegenseitige TLS für einen benutzerdefinierten Domännennamen aktivieren und dann den benutzerdefinierten Domännennamen für Clients bereitstellen. Um mithilfe eines benutzerdefinierten Domännennamens auf eine API zuzugreifen, für die gegenseitige TLS aktiviert ist, müssen Clients Zertifikate präsentieren,

denen Sie in API-Anforderungen vertrauen. Sie finden mehr Informationen in [the section called “Benutzerdefinierte Domainnamen”](#).

Verwendung AWS Certificate Manager ausgestellter Zertifikate

Sie können ein öffentlich vertrauenswürdigen Zertifikat direkt von ACM anfordern oder öffentliche oder selbstsignierte Zertifikate importieren. Gehen Sie zum Einrichten eines Zertifikats in ACM zu [ACM](#). Wenn Sie ein Zertifikat importieren möchten, lesen Sie weiter im folgenden Abschnitt.

Verwenden eines importierten AWS Private Certificate Authority Zertifikats oder Zertifikats

Um ein in ACM importiertes Zertifikat oder ein Zertifikat von Mutual TLS verwenden AWS Private Certificate Authority zu können, benötigt API Gateway ein von ACM ownershipVerificationCertificate ausgestelltes. Dieses Eigentumszertifikat wird nur verwendet, um zu überprüfen, ob Sie über Berechtigungen zur Verwendung des Domänennamens verfügen. Es wird nicht für den TLS-Handshake verwendet. Falls Sie noch keinen habenownershipVerificationCertificate, gehen Sie zu, <https://console.aws.amazon.com/acm/>um einen einzurichten.

Sie müssen dieses Zertifikat für die gesamte Lebensdauer Ihres Domainnamens gültig halten. Wenn ein Zertifikat abläuft und die automatische Verlängerung fehlschlägt, werden alle Updates des Domänennamens gesperrt. Sie müssen die ownershipVerificationCertificateArn mit einem gültigen ownershipVerificationCertificate aktualisieren, bevor Sie weitere Änderungen vornehmen können. Die ownershipVerificationCertificate kann nicht als Serverzertifikat für eine andere gemeinsame TLS-Domäne in API Gateway verwendet werden. Wenn ein Zertifikat direkt wieder in ACM importiert wird, muss der Aussteller gleich bleiben.

Konfigurieren Ihres Truststore

Truststores sind Textdateien mit einer .pem-Dateierweiterung. Sie sind eine vertrauenswürdige Liste von Zertifikaten von Zertifizierungsstellen. Um Mutual TLS zu verwenden, erstellen Sie einen Truststore mit X.509-Zertifikaten, denen Sie beim Zugriff auf Ihre API vertrauen.

Sie müssen die vollständige Vertrauenskette vom ausstellenden CA-Zertifikat bis zum Stamm-CA-Zertifikat in Ihren Truststore aufnehmen. API Gateway akzeptiert Clientzertifikate, die von jeder Zertifizierungsstelle in der Vertrauenskette ausgestellt werden. Die Zertifikate können von öffentlichen oder privaten Zertifizierungsstellen stammen. Zertifikate können eine maximale Kettenlänge von vier haben. Sie können auch selbstsignierte Zertifikate bereitstellen. Die folgenden Hash-Algorithmen werden im Truststore unterstützt:

- SHA-256 oder stärker
- RSA-2048 oder stärker
- ECDSA-256 oder stärker

API Gateway validiert eine Reihe von Zertifikatseigenschaften. Sie können Lambda-Genehmiger verwenden, um zusätzliche Prüfungen durchzuführen, wenn ein Client eine API aufruft, einschließlich der Prüfung, ob ein Zertifikat widerrufen wurde. API Gateway validiert die folgenden Eigenschaften:

Validierung	Beschreibung
X.509-Syntax	Das Zertifikat muss die X.509-Syntaxanforderungen erfüllen.
Integrität	Der Inhalt des Zertifikats darf sich nicht von dem unterscheiden, was von der Zertifizierungsstelle des Truststore signiert wurde.
Gültigkeit	Die Gültigkeitsdauer des Zertifikats muss aktuell sein.
Namensverkettung/Schlüsselverkettung	Die Verkettung der Namen und Antragssteller von Zertifikaten darf nicht unterbrochen sein. Zertifikate können eine maximale Kettenlänge von vier haben.

Hochladen des Vertrauensspeichers in einen Amazon-S3-Bucket in einer einzigen Datei.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
```

...

Der folgende [AWS CLI cp-Befehl](#) lädt `certificates.pem` in Ihren Amazon S3 S3-Bucket hoch:

```
aws s3 cp certificates.pem s3://bucket-name
```

Konfigurieren von gegenseitiger TLS für einen benutzerdefinierten Domännennamen

Um gegenseitiges TLS für eine HTTP-API zu konfigurieren, müssen Sie einen benutzerdefinierten regionalen Domännennamen und mindestens TLS-Version 1.2 für Ihre API verwenden. Weitere Informationen zum Erstellen und Konfigurieren eines benutzerdefinierten Domännennamens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

Note

Mutual TLS wird für private APIs Zwecke nicht unterstützt.

Nachdem Sie Ihren Vertrauensspeicher in Amazon S3 hochgeladen haben, können Sie Ihren benutzerdefinierten Domännennamen für die Verwendung von gegenseitigem TLS konfigurieren. Im Folgenden [create-domain-name](#) wird ein benutzerdefinierter Domainname mit gegenseitigem TLS erstellt:

```
aws apigatewayv2 create-domain-name \  
  --domain-name api.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/123456789012-1234-1234-12345678 \  
  --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Nachdem Sie den Domännennamen erstellt haben, müssen Sie DNS-Datensätze und Basepath-Mappings für API-Vorgänge konfigurieren. Weitere Informationen hierzu finden Sie unter [Einrichten eines regionalen benutzerdefinierten Domainnamens in API Gateway](#).

Aufrufen einer API mithilfe eines benutzerdefinierten Domännennamens, der gegenseitige TLS erfordert

Um eine API mit aktivierter gegenseitiger TLS aufzurufen, müssen Clients in der API-Anforderung ein vertrauenswürdiges Zertifikat präsentieren. Wenn ein Client versucht, Ihre API aufzurufen, sucht API Gateway in Ihrem Truststore nach dem Aussteller des Clientzertifikats. Damit API Gateway mit

der Anforderung fortfahren kann, müssen sich der Aussteller des Zertifikats und die vollständige Vertrauenskette bis zum Stamm-CA-Zertifikats in Ihrem Truststore befinden.

Mit dem folgenden `curl`-Beispielbefehl wird eine Anforderung an `api.example.com`, gesendet, die `my-cert.pem` in die Anforderung einschließt. `my-key.key` ist der private Schlüssel für das Zertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Ihre API wird nur aufgerufen, wenn Ihr Truststore dem Zertifikat vertraut. Die folgenden Bedingungen führen dazu, dass API Gateway den TLS-Handshake fehlschlägt und die Anforderung mit einem 403-Statuscode verweigert. Wenn Ihr Zertifikat:

- nicht vertrauenswürdig ist
- abgelaufen ist
- keinen unterstützten Algorithmus verwendet

Note

API Gateway überprüft nicht, ob ein Zertifikat widerrufen wurde.

Aktualisieren Ihres Truststore

Um die Zertifikate in Ihrem Truststore zu aktualisieren, laden Sie ein neues Zertifikatspaket in Amazon S3 hoch. Anschließend können Sie Ihren benutzerdefinierten Domännennamen aktualisieren, um das aktualisierte Zertifikat zu verwenden.

Verwenden Sie [Amazon-S3-Versioning](#), um mehrere Versionen Ihres Vertrauensspeichers zu verwalten. Wenn Sie Ihren benutzerdefinierten Domännennamen aktualisieren, um eine neue Vertrauensspeicher-Version zu verwenden, gibt API Gateway Warnungen zurück, wenn Zertifikate ungültig sind.

API Gateway erzeugt Zertifikatswarnungen nur dann, wenn Sie Ihren Domännennamen aktualisieren. API Gateway benachrichtigt Sie nicht, wenn ein zuvor hochgeladenes Zertifikat abläuft.

Mit dem folgenden [update-domain-name](#)Befehl wird ein benutzerdefinierter Domainname aktualisiert, sodass er eine neue Truststore-Version verwendet:

```
aws apigatewayv2 update-domain-name \  
  --domain-name api.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-12345678 \  
  --mutual-tls-authentication TruststoreVersion='abcdef123'
```

Gegenseitige TLS deaktivieren

Um gegenseitige TLS für einen benutzerdefinierten Domännennamen zu deaktivieren, entfernen Sie den Truststore aus Ihrem benutzerdefinierten Domännennamen, wie im folgenden Befehl gezeigt.

Der folgende [update-domain-name](#) Befehl aktualisiert einen benutzerdefinierten Domainnamen, um den Truststore aus Ihrem benutzerdefinierten Domainnamen zu entfernen:

```
aws apigatewayv2 update-domain-name \  
  --domain-name api.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-12345678 \  
  --mutual-tls-authentication TruststoreUri=''
```

Problembehebung bei gegenseitigen TLS für Ihre HTTP-API

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Aktivierung von „Gegenseitige TLS“ auftreten können.

Fehlerbehebung bei Zertifikatwarnungen

Wenn Sie einen benutzerdefinierten Domännennamen mit gegenseitiger TLS erstellen, gibt API Gateway Warnungen zurück, wenn Zertifikate im Truststore ungültig sind. Dies kann auch auftreten, wenn ein benutzerdefinierter Domännennamen aktualisiert wird, um einen neuen Truststore zu verwenden. Die Warnungen geben das Problem mit dem Zertifikat und den Betreff des Zertifikats an, das die Warnung erstellt hat. Gegenseitige TLS ist weiterhin für Ihre API aktiviert, aber einige Clients können möglicherweise nicht auf Ihre API zugreifen.

Um aufzudecken, welches Zertifikat die Warnung erzeugt hat, dekodieren Sie die Zertifikate in Ihrem Truststore. Sie können Tools wie `openssl` zum Dekodieren der Zertifikate und zur Identifizierung ihrer Subjekte verwenden.

Der folgende Befehl zeigt den Inhalt eines Zertifikats einschließlich seines Betreffs an:


```
openssl x509 -in certificate.crt -text -noout
```

Aktualisieren oder entfernen Sie die Zertifikate, die Warnungen erstellt haben, und laden Sie dann einen neuen Truststore in Amazon S3 hoch. Nachdem Sie einen neuen Truststore hochgeladen haben, aktualisieren Sie Ihren benutzerdefinierten Domännennamen, um den neuen Truststore zu verwenden.

Problembehandlung bei Domännennamenskonflikten

Der Fehler "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." bedeutet, dass mehrere Zertifizierungsstellen ein Zertifikat für diese Domäne ausgestellt haben. Für jeden Subjekt im Zertifikat kann es nur einen Aussteller in API Gateway für gegenseitige TLS-Domänen geben. Sie müssen alle Ihre Zertifikate für dieses Thema über einen einzigen Aussteller abrufen. Wenn das Problem mit einem Zertifikat besteht, das Sie nicht kontrollieren können, Sie jedoch den Besitz des Domännennamens nachweisen können, [kontaktieren Sie Support](#), um ein Ticket zu eröffnen.

Fehlerbehebung bei Statusmeldungen für Domännennamen

PENDING_CERTIFICATE_REIMPORT: Das bedeutet, dass Sie ein Zertifikat erneut in ACM importiert haben und die Validierung fehlgeschlagen ist, weil das neue Zertifikat einen SAN (Subject Alternative Name) hat, der nicht durch den ownershipVerificationCertificate oder der Betreff abgedeckt ist oder der Betreff oder SANs im Zertifikat nicht den Domainnamen abdeckt. Möglicherweise ist etwas falsch konfiguriert oder ein ungültiges Zertifikat wurde importiert. Sie müssen ein gültiges Zertifikat erneut in ACM importieren. Weitere Informationen zur Validierung finden Sie unter [Validierung des Domänenbesitzes](#).

PENDING_OWNERSHIP_VERIFICATION: Dies bedeutet, dass Ihr zuvor verifiziertes Zertifikat abgelaufen ist und ACM es nicht automatisch erneuern konnte. Sie müssen das Zertifikat erneuern oder ein neues Zertifikat anfordern. Weitere Informationen zur Zertifikatserneuerung finden Sie im Leitfaden [ACMs Fehlerbehebung bei der Erneuerung verwalteter Zertifikate](#).

Überwachen Sie HTTP APIs im API Gateway

Sie können CloudWatch Metriken und CloudWatch Logs verwenden, um HTTP zu überwachen APIs. Durch die Kombination von Protokollen und Metriken können Sie Fehler protokollieren und die Leistung Ihrer API überwachen.

 Note

API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:

- 413 Request Entity Too Large-Fehler
- Übermäßige 429 Too Many Requests-Fehler
- Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
- Fehler der Serie 500, die durch interne Ausfälle verursacht werden

Themen

- [CloudWatch-Metriken für HTTP-APIs in API Gateway überwachen](#)
- [Konfigurieren Sie die Protokollierung für HTTP APIs in API Gateway](#)

CloudWatch-Metriken für HTTP-APIs in API Gateway überwachen

Sie können die API-Ausführung mithilfe von CloudWatch überwachen. Dabei werden Rohdaten von API Gateway gesammelt und zu lesbaren, nahezu Echtzeitmetriken verarbeitet. Diese Statistiken werden für einen Zeitraum von 15 Monaten aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt wird. Standardmäßig werden API-Gateway-Metrikdaten automatisch in Intervallen von einer Minute an CloudWatch gesendet. Um Ihre Metriken zu überwachen, erstellen Sie ein CloudWatch-Dashboard für Ihre API. Weitere Informationen zum Erstellen eines CloudWatch-Dashboards finden Sie unter [Erstellen eines CloudWatch-Dashboards](#) im Amazon-CloudWatch-Benutzerhandbuch. Weitere Informationen finden Sie unter [Was ist Amazon CloudWatch?](#) im Amazon CloudWatch-Benutzerhandbuch.

Die folgenden Metriken werden für HTTP-APIs unterstützt. Sie können außerdem detaillierte Metriken aktivieren, um Metriken auf Routenebene in Amazon CloudWatch zu schreiben.

Metrik	Beschreibung
4xx	Die Anzahl der erfassten clientseitigen Fehler innerhalb eines bestimmten Zeitraums.
5xx	Die Anzahl der erfassten serverseitigen Fehler innerhalb eines bestimmten Zeitraums
Anzahl	Die Gesamtzahl der API-Anforderungen innerhalb eines bestimmten Zeitraums
IntegrationLatency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage an das Backend weiterleitet und zu dem Zeitpunkt, zu dem es eine Antwort vom Backend erhält.
Latency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage von einem Client empfängt, und dem Zeitpunkt, an dem es eine Antwort an den Client zurückgibt. Die Latenzzeit schließt die Integrationslatenzzeit und anderen API Gateway-Overhead ein.
DataProcessed	Die Menge der verarbeiteten Daten in Bytes.

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

Dimension	Beschreibung
Apild	Filtert API Gateway-Metriken für eine API mit der angegebenen API-ID.
Apild, Stage	Filtert API Gateway-Metriken für eine API-Stufe mit der angegebenen API-ID und Stufen-ID.
Apild, Methode, Ressource, Stufe	Filtert API-Gateway-Metriken für eine API-Methode mit der angegebenen API-ID, Stufen-ID, dem Ressourcenpfad und der Routen-ID. API Gateway sendet diese Metriken nur dann, wenn Sie detaillierte CloudWatch-Metriken explizit aktiviert haben. Hierzu rufen Sie die Aktion UpdateStage der API-Gateway-V2-REST-API auf, um die Eigenschaft <code>detailedMetricsEnabled</code> in <code>true</code> zu aktualisieren. Alternativ können Sie den AWS CLI-Befehl update-stage aufrufen, um die <code>DetailedMetricsEnabled</code>-Eigenschaft zu <code>true</code> zu aktualisieren. Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen

Dimension	Beschreibung
	Gebühren belastet. Preisinformationen finden Sie unter Amazon CloudWatch-Preisinformationen .

Konfigurieren Sie die Protokollierung für HTTP APIs in API Gateway

Sie können die Protokollierung aktivieren, um Protokolle in Logs zu CloudWatch schreiben. Sie können [Protokollierungsvariablen](#) verwenden, um den Inhalt Ihrer Protokolle anzupassen.

Um Ihre Sicherheitslage zu verbessern, empfehlen wir Ihnen, für alle Phasen Ihrer HTTP-API CloudWatch Protokolle in Logs zu schreiben. Dies kann erforderlich sein, um verschiedene Compliance-Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

Um die Protokollierung für eine HTTP-API einzuschalten, müssen Sie Folgendes tun.

1. Stellen Sie sicher, dass Ihr -Benutzer über die erforderlichen Berechtigungen zum Einschalten der Protokollierung verfügt.
2. Erstellen Sie eine CloudWatch Protokollgruppe für Logs.
3. Geben Sie den ARN der CloudWatch Logs-Protokollgruppe für eine Phase Ihrer API an.

Berechtigungen zum Einschalten der Protokollierung

Um die Protokollierung für eine API einzuschalten, muss der -Benutzer über die folgenden Berechtigungen verfügen.

Example

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
        "Action": [
            "logs:DescribeLogGroups",
            "logs:DescribeLogStreams",
            "logs:GetLogEvents",
            "logs:FilterLogEvents"
        ],
        "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogDelivery",
            "logs:PutResourcePolicy",
            "logs:UpdateLogDelivery",
            "logs>DeleteLogDelivery",
            "logs:CreateLogGroup",
            "logs:DescribeResourcePolicies",
            "logs:GetLogDelivery",
            "logs:ListLogDeliveries"
        ],
        "Resource": "*"
    }
]
```

Erstellen Sie eine Protokollgruppe und aktivieren Sie die Protokollierung für HTTP APIs

Sie können eine Protokollgruppe erstellen und die Zugriffsprotokollierung mit dem AWS-Managementkonsole oder dem aktivieren AWS CLI.

AWS-Managementkonsole

1. Erstellen Sie eine -Protokollgruppe.

Informationen zum Erstellen einer Protokollgruppe mithilfe der Konsole finden Sie unter [Erstellen einer Protokollgruppe im Amazon CloudWatch Logs-Benutzerhandbuch](#).

2. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway>.
3. Wählen Sie eine HTTP-API.

4. Wählen Sie unter der Registerkarte Monitor (Überwachen) im Hauptnavigationsbereich die Option Logging (Protokollierung) aus.
5. Wählen Sie eine Phase aus, um die Protokollierung zu aktivieren, und wählen Sie Select (Auswählen) aus.
6. Wählen Sie Edit (Bearbeiten) aus, um die Zugriffsprotokollierung zu aktivieren.
7. Aktivieren Sie die Zugriffsprotokollierung, geben Sie ein CloudWatch Protokoll ein und wählen Sie ein Protokollformat aus.
8. Wählen Sie Speichern.

AWS CLI

Mit dem folgenden [create-log-group](#) Befehl wird eine Protokollgruppe erstellt:

```
aws logs create-log-group --log-group-name my-log-group
```

Zum Einschalten der Protokollierung benötigen Sie den Amazon-Ressourcennamen (ARN) für Ihre Protokollgruppe. Das ARN-Format ist `arn:aws:logs::log-group:region account-id log-group-name`

Der folgende [update-stage](#) Befehl aktiviert die Protokollierung für die `$default`-Stufe einer HTTP-API.

```
aws apigatewayv2 update-stage --api-id abcdef \  
  --stage-name '$default' \  
  --access-log-settings '{"DestinationArn": "arn:aws:logs:region:account-id:log-group:log-group-name", "Format": "$context.identity.sourceIp - -  
[$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\"  
$context.status $context.responseLength $context.requestId"}'
```

Beispielprotokollformate

Beispiele für einige gängige Zugriffsprotokollformate sind in der API Gateway-Konsole verfügbar und im Folgenden aufgeführt.

- CLF ([Common Log Format](#)):

```
$context.identity.sourceIp - - [$context.requestTime] "$context.httpMethod
$context.routeKey $context.protocol" $context.status $context.responseLength
$context.requestId $context.extendedRequestId
```

- JSON:

```
{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
  "requestTime":"$context.requestTime",
  "httpMethod":"$context.httpMethod","routeKey":"$context.routeKey",
  "status":"$context.status","protocol":"$context.protocol",
  "responseLength":"$context.responseLength", "extendedRequestId":
  "$context.extendedRequestId" }
```

- XML:

```
<request id="$context.requestId"> <ip>$context.identity.sourceIp</ip> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
  <routeKey>$context.routeKey</routeKey> <status>$context.status</status> <protocol>
$context.protocol</protocol> <responseLength>$context.responseLength</responseLength>
  <extendedRequestId>$context.extendedRequestId</extendedRequestId> </request>
```

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

HTTP-API-Zugriffsprotokolle anpassen

Sie können die folgenden Variablen zur Anpassung der HTTP-API-Zugriffsprotokolle verwenden. Weitere Informationen zu Zugriffsprotokollen für HTTP APIs finden Sie unter [Konfigurieren Sie die Protokollierung für HTTP APIs in API Gateway](#).

Parameter	Description
<code>\$context.accountId</code>	Die AWS Konto-ID des API-Besitzers.
<code>\$context.apiId</code>	Die ID, die API Gateway Ihrer API zuweist.

Parameter	Description
<code>\$context.authorizer.claims. <i>property</i></code>	Eine Eigenschaft der Ansprüche, die das JSON Web Token (JWT) zurückgibt, nachdem der Aufrufer der Methode erfolgreich authentifiziert wurde (z. B. <code>\$context.authorizer.claims.username</code>). Weitere Informationen finden Sie unter Steuern Sie den Zugriff auf HTTP APIs mit JWT-Autorisierern in API Gateway.  Note Bei einem Aufruf von <code>\$context.authorizer.claims</code> wird null (0) zurückgegeben.
<code>\$context.authorizer.error</code>	Die von einem Genehmiger zurückgegebene Fehlermeldung.

Parameter	Description
<code>\$context.authorizer.</code> <i>property</i>	Der Wert des angegebenen Schlüssel-Wert-Paares der context-Zuordnung, die von einer Lambda-Genehmigerfunktion des API Gateways zurückgegeben wird. Angenommen, der Genehmiger gibt folgende context-Zuweisung zurück: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Der Aufruf von <code>\$context.authorizer.key</code> gibt die Zeichenfolge "value" zurück, der Aufruf von <code>\$context.authorizer.numKey</code> gibt 1 zurück, und der Aufruf von <code>\$context.authorizer.boolKey</code> gibt true zurück.
<code>\$context.awsEndpointRequestId</code>	Die Anforderungs-ID des AWS Endpunkts aus dem <code>x-amz-request-id</code> <code>x-amzn-requestId</code> OR-Header.
<code>\$context.awsEndpointRequestId2</code>	Die Anforderungs-ID des AWS Endpunkts aus dem <code>x-amz-id-2</code> Header.

Parameter	Description
<code>\$context.customDomain.basePathMatched</code>	Der Pfad für ein API-Mapping, mit dem eine eingehende Anforderung übereinstimmt. Gilt, wenn ein Client einen benutzerdefinierten Domain-Namen für den Zugriff auf eine API verwendet. Wenn ein Client beispielsweise eine Anforderung an <code>https://api.example.com/v1/orders/1234</code> sendet und die Anforderung dem API-Mapping mit dem Pfad <code>v1/orders</code> übereinstimmt, lautet der Wert <code>v1/orders</code> . Weitere Informationen hierzu finden Sie unter the section called “API-Zuweisungen” .
<code>\$context.dataProcessed</code>	Die Menge der verarbeiteten Daten in Bytes.
<code>\$context.domainName</code>	Der zum Aufrufen der API verwendete vollständige Domänenname. Dieser Wert sollte mit dem für den eingehenden Host-Header übereinstimmen.
<code>\$context.domainPrefix</code>	Das erste Label der <code>\$context.domainName</code> .
<code>\$context.error.message</code>	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält.
<code>\$context.error.messageString</code>	Der Wert von <code>\$context.error.message</code> in Anführungszeichen, d. h. <code>"\$context.error.message"</code> .
<code>\$context.error.responseType</code>	Ein Typ von <code>GatewayResponse</code> . Weitere Informationen erhalten Sie unter the section called “Kennzahlen” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen” .

Parameter	Description
<code>\$context.extendedRequestId</code>	Äquivalent mit <code>\$context.requestId</code> .
<code>\$context.httpMethod</code>	Die verwendete HTTP-Methode. Gültige Werte sind: DELETE, GET, HEAD, OPTIONS, PATCH, POST und PUT.
<code>\$context.identity.accountId</code>	Die der Anfrage zugeordnete AWS Konto-ID. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.caller</code>	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.cognitoAuthenticationProvider</code>	Eine durch Komma getrennte Liste aller Amazon-Cognito-Authentifizierungsanbieter, die vom anfordernden Aufrufer verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, <code>cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Weitere Informationen zu verfügbaren Amazon-Cognito-Authentifizierungsanbietern finden Sie unter Verbundidentitäten verwenden im Amazon-Cognito-Entwicklerhandbuch.

Parameter	Description
<code>\$context.identity.cognitoAuthenticationType</code>	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind <code>authenticated</code> für authentifizierte Identitäten und <code>unauthenticated</code> für nicht authentifizierte Identitäten.
<code>\$context.identity.cognitoIdentityId</code>	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.cognitoIdentityPoolId</code>	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.principalOrgId</code>	Die AWS -Organisations-ID . Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Das PEM-codierte Clientzertifikat, das der Client während der gegenseitigen TLS-Authentifizierung präsentiert hat. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domännennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.subjectDN</code>	Der Distinguished Name des Zertifikatantrags, den ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domännennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.

Parameter	Description
<code>\$context.identity.clientCertificate.issuerDN</code>	Der Distinguished Name des Ausstellers des Zertifikats, das ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.serialNumber</code>	Die Seriennummer des Zertifikats. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Das Datum, vor dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Das Datum, nach dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
<code>\$context.identity.sourceIp</code>	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
<code>\$context.identity.user</code>	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.userAgent</code>	Die User-Agent -Kopfzeile des API-Aufrufers.

Parameter	Description
<code>\$context.identity.userArn</code>	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.integration.error</code>	Die von einer Integration zurückgegebene Fehlermeldung. Äquivalent mit <code>\$context.integrationErrorMessage</code> .
<code>\$context.integration.integrationStatus</code>	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.
<code>\$context.integration.latency</code>	Die Integrationslatenz in Millisekunden. Äquivalent mit <code>\$context.integrationLatency</code> .
<code>\$context.integration.requestId</code>	Die AWS Anforderungs-ID des Endpunkts. Äquivalent mit <code>\$context.awsEndpointRequestId</code> .
<code>\$context.integration.status</code>	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird.
<code>\$context.integrationErrorMessage</code>	Eine Zeichenfolge, die eine Integrationsfehlermeldung enthält.
<code>\$context.integrationLatency</code>	Die Integrationslatenz in Millisekunden.

Parameter	Description
<code>\$context.integrationStatus</code>	Bei der Lambda-Proxyintegration stellt dieser Parameter den Statuscode dar, der von der Backend-Lambda-Funktion zurückgegeben wird AWS Lambda, nicht von der Backend-Funktion.
<code>\$context.path</code>	Der Anforderungspfad. Beispiel, <code>/[stage]/root/child</code> .
<code>\$context.protocol</code>	Das Anforderungsprotokoll ist z. B. HTTP/1.1.  Note API Gateway APIs kann HTTP/2-Anfragen akzeptieren, aber API Gateway sendet Anfragen über HTTP/1.1 an Backend-Integrationen. Infolgedessen wird das Anforderungsprotokoll als HTTP/1.1 protokolliert, auch wenn ein Client eine Anfrage sendet, die HTTP/2 verwendet.
<code>\$context.requestId</code>	Die ID, die API Gateway der API-Anfrage zuweist.
<code>\$context.requestTime</code>	Die Anforderungszeit im CLF -Format (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	Die Anforderungszeit im Epoch -Format.
<code>\$context.responseLatency</code>	Die Antwortlatenz in Millisekunden.
<code>\$context.responseLength</code>	Die Länge der Antwortnutzlast in Byte.
<code>\$context.routeKey</code>	Der Routing-Schlüssel der API-Anfrage, z. B. <code>/pets</code> .

Parameter	Description
<code>\$context.stage</code>	Die Bereitstellungsstufe der API-Anforderung (z. B. beta oder prod).
<code>\$context.status</code>	Der Status der Methodenantwort.

Fehlerbehebung bei Problemen mit HTTP-APIs in API Gateway

Die folgenden Themen enthalten Ratschläge zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung von HTTP-APIs auftreten können.

Themen

- [Fehlerbehebung bei Problemen mit HTTP-API-Lambda-Integrationen](#)
- [Fehlerbehebung bei Problemen mit HTTP-API JWT-Genehmigern](#)

Fehlerbehebung bei Problemen mit HTTP-API-Lambda-Integrationen

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung [AWS Lambda Integrationen](#) mit HTTP APIs auftreten können.

Problem: Meine API mit einer Lambda-Integration gibt zurück

`{"message": "Internal Server Error"}`

Um den internen Serverfehler zu beheben, fügen Sie die `$context.integrationErrorMessage` [Protokollierungsvariable](#) zur Ihrem Protokollformat hinzu und sehen Sie sich die Protokolle Ihrer HTTP-API an. Um dies zu erreichen, gehen Sie wie folgt vor:

Um eine Protokollgruppe zu erstellen, verwenden Sie AWS-Managementkonsole

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie Protokollgruppen.
3. Wählen Sie Protokollgruppe erstellen.
4. Geben Sie einen Protokollgruppennamen ein und wählen Sie dann Erstellen.

5. Notieren Sie sich den Amazon-Ressourcennamen (ARN) für Ihre Protokollgruppe. Das ARN-Format ist `arn:aws:logs:::log-group:: region account-id log-group-name`. Sie benötigen den ARN der Protokollgruppe, um die Zugriffsprotokollierung für Ihre HTTP-API zu aktivieren.

So fügen Sie die `$context.integrationErrorMessage`-Protokollierungsvariable hinzu

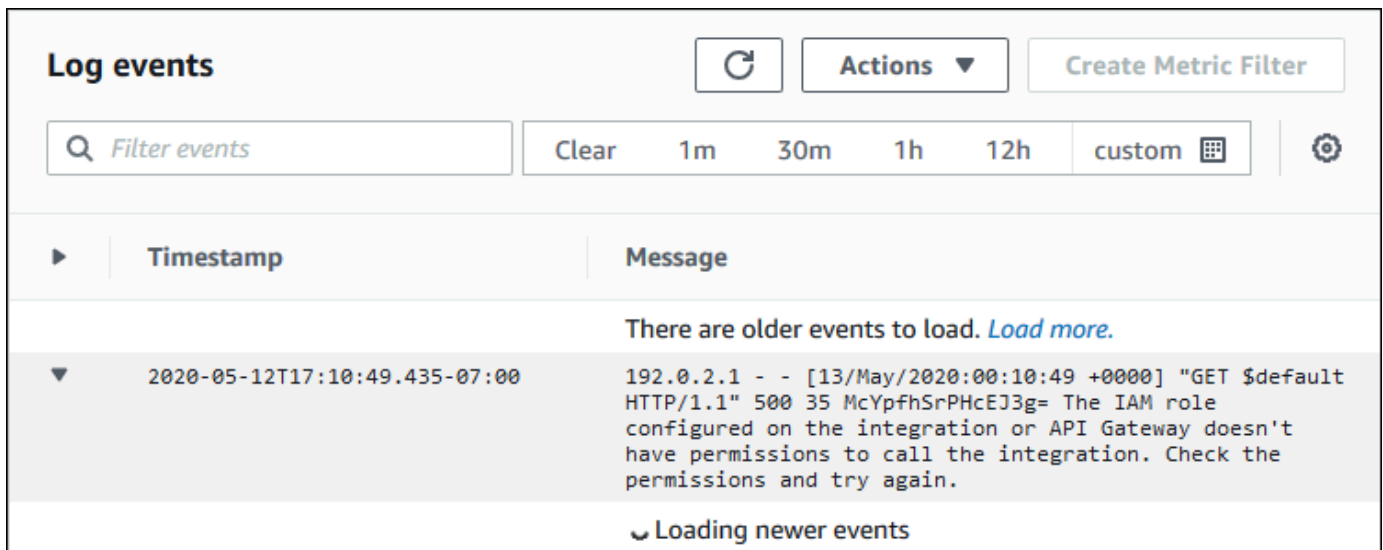
1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre HTTP-API.
3. Wählen Sie unter Monitor (Überwachen) die Option Logging (Protokollierung).
4. Wählen Sie eine Stufe Ihrer API aus.
5. Wählen Sie Bearbeiten und aktivieren Sie dann die Zugriffsprotokollierung.
6. Geben Sie für Protokollstandort den ARN der Protokollgruppe ein, die Sie im vorherigen Schritt erstellt haben.
7. Wählen Sie für Protokollformat die Option CLF aus. API Gateway erstellt ein Beispielprotokollformat.
8. Fügen Sie `$context.integrationErrorMessage` am Ende des Protokollformats hinzu.
9. Wählen Sie Save (Speichern) aus.

So zeigen Sie die Protokolle Ihrer API an

1. Generieren von Protokollen. Verwenden Sie einen Browser oder `curl`, um Ihre API aufzurufen.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
3. Wählen Sie Ihre HTTP-API.
4. Wählen Sie unter Monitor (Überwachen) die Option Logging (Protokollierung).
5. Wählen Sie die Stufe Ihrer API aus, für die Sie die Protokollierung aktiviert haben.
6. Wählen Sie „Logs in anzeigen“. CloudWatch
7. Wählen Sie den neuesten Protokoll-Stream, um die Protokolle Ihrer HTTP-API einzusehen.
8. Ihr Protokolleintrag sollte wie folgt aussehen:



Da wir `$context.integrationErrorMessage` dem Protokollformat hinzugefügt haben, wird in unseren Protokollen eine Fehlermeldung angezeigt, die das Problem zusammenfasst.

Ihre Protokolle enthalten möglicherweise eine andere Fehlermeldung, die darauf hinweist, dass ein Problem mit Ihrem Lambda-Funktionscode vorliegt. Überprüfen Sie in diesem Fall Ihren Lambda-Funktionscode und vergewissern Sie sich, dass Ihre Lambda-Funktion eine Antwort im [erforderlichen Format](#) zurückgibt. Wenn Ihre Protokolle keine Fehlermeldung enthalten, fügen Sie `$context.error.message` und `$context.error.responseType` Ihrem Protokollformat hinzu, um weitere Informationen zur Fehlerbehebung zu erhalten.

In diesem Fall zeigen die Protokolle, dass API Gateway nicht über die erforderlichen Berechtigungen zum Aufrufen der Lambda-Funktion verfügte.

Wenn Sie eine Lambda-Integration in der API Gateway-Konsole erstellen, konfiguriert API Gateway automatisch die Berechtigungen zum Aufrufen der Lambda-Funktion. Wenn Sie eine Lambda-Integration mithilfe des AWS CLI CloudFormation, oder eines SDK erstellen, müssen Sie API Gateway Berechtigungen zum Aufrufen der Funktion gewähren. Die folgenden [add-permission](#)-Befehle erteilen die Berechtigung für verschiedene HTTP-API-Routen, eine Lambda-Funktion aufzurufen.

Example Beispiel – Für die `$default`-Stufe und die `$default`-Route einer HTTP-API

```
aws lambda add-permission \
  --function-name my-function \
  --statement-id apigateway-invoke-permissions \
  --action lambda:InvokeFunction \
```

```
--principal apigateway.amazonaws.com \
--source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/\$default/\$default"
```

Example Beispiel – Für die **prod**-Stufe und die **test**-Route einer HTTP-API

```
aws lambda add-permission \
--function-name my-function \
--statement-id apigateway-invoke-permissions \
--action lambda:InvokeFunction \
--principal apigateway.amazonaws.com \
--source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/prod/*/test"
```

[Bestätigen Sie die Funktionsrichtlinie](#) auf der Registerkarte Permissions (Berechtigungen) der Lambda-Konsole.

Versuchen Sie erneut, Ihre API aufzurufen. Sie sollten die Antwort Ihrer Lambda-Funktion sehen.

Fehlerbehebung bei Problemen mit HTTP-API JWT-Genehmigern

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung von JSON Web Token (JWT) -Autorisierern mit HTTP auftreten können. APIs

Problem: Meine API gibt zurück 401 {"message":"Unauthorized"}

Überprüfen Sie den `www-authenticate`-Header in der Antwort von der API.

Der folgende Befehl verwendet `curl`, um eine Anforderung an eine API mit einem JWT-Genehmiger zu senden, der `$request.header.Authorization` als Identitätsquelle verwendet.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-west-2.amazonaws.com/route
```

Die Antwort von der API enthält einen `www-authenticate`-Header.

```
...
< HTTP/1.1 401 Unauthorized
< Date: Wed, 13 May 2020 04:07:30 GMT
< Content-Length: 26
< Connection: keep-alive
```

```
< www-authenticate: Bearer scope="" error="invalid_token" error_description="the token
  does not have a valid audience"
< apigw-requestid: Mc7UVioPPHcEKPA=
<
* Connection #0 to host api-id.execute-api.us-west-2.amazonaws.com left intact
{"message":"Unauthorized"}}
```

In diesem Fall zeigt der `www-authenticate`-Header an, dass das Token nicht für eine gültige Zielgruppe ausgegeben wurde. Damit API Gateway eine Anfrage autorisieren kann, muss der `aud`- oder `client_id`-Claim von JWT mit einem der Audience-Einträge übereinstimmen, die für den Genehmiger konfiguriert sind. API Gateway validiert `client_id` nur dann, wenn `aud` nicht vorhanden ist. Wenn sowohl `aud` als auch `client_id` vorhanden sind, bewertet API Gateway `aud`.

Sie können auch ein JWT dekodieren und überprüfen, ob es mit dem Aussteller, der Zielgruppe und den Bereichen übereinstimmt, die Ihre API benötigt. Die Website jwt.io kann im Browser debuggen. JWTs Die OpenID Foundation führt eine [Liste von Bibliotheken, mit JWTs denen Sie arbeiten](#) können.

Weitere Informationen zu JWT-Genehmigern finden Sie unter [Steuern Sie den Zugriff auf HTTP APIs mit JWT-Autorisierern in API Gateway](#).

API-Gateway-WebSocket-API

Eine WebSocket-API in API Gateway ist eine Sammlung von WebSocket-Routen, die in HTTP-Endpunkte, Lambda-Funktionen oder andere AWS-Services im Backend integriert sind. Sie können API Gateway-Funktionen verwenden, um Ihnen bei allen Aspekten des API-Lebenszyklus zu helfen, von der Erstellung bis zur Überwachung Ihrer Produktions-APIs.

API Gateway-WebSocket-APIs sind bidirektional. Ein Client kann Nachrichten an einen Service senden und Services können unabhängig Nachrichten an Clients senden. Dieses bidirektionale Verhalten ermöglicht umfassendere Client/Service-Interaktionen, da Services Daten an Clients übertragen können, ohne dass Clients eine explizite Anforderung stellen müssen. WebSocket-APIs werden häufig für Echtzeit-Anwendungen wie Chat-Anwendungen, Plattformen für die Zusammenarbeit, Multiplayer-Spiele und Plattformen für Finanztransaktionen verwendet.

Eine Beispiel-App, mit der Sie beginnen können, finden Sie unter [Tutorial: Erstellen Sie eine WebSocket Chat-App mit einer WebSocket API, Lambda und DynamoDB](#).

In diesem Abschnitt erfahren Sie, wie Sie Ihre WebSocket-APIs mit API Gateway entwickeln, veröffentlichen, schützen und überwachen.

Themen

- [Überblick über WebSocket APIs in API Gateway](#)
- [WebSocket APIs In API Gateway entwickeln](#)
- [WebSocket-APIs für Kundenaufrufe veröffentlichen](#)
- [Schützen Sie Ihr WebSocket APIs internes API Gateway](#)
- [WebSocket-APIs in API Gateway überwachen](#)

Überblick über WebSocket APIs in API Gateway

In API Gateway können Sie eine WebSocket API als Stateful-Frontend für einen AWS Dienst (wie Lambda oder DynamoDB) oder für einen HTTP-Endpunkt erstellen. Die WebSocket API ruft Ihr Backend auf der Grundlage des Inhalts der Nachrichten auf, die sie von Client-Apps empfängt.

Im Gegensatz zu einer REST-API, die Anfragen empfängt und beantwortet, unterstützt eine WebSocket API die bidirektionale Kommunikation zwischen Client-Apps und Ihrem Backend. Das Backend kann Rückruf-Nachrichten an verbundene Clients senden.

In Ihrer WebSocket API werden eingehende JSON-Nachrichten auf der Grundlage von Routen, die Sie konfigurieren, an Backend-Integrationen weitergeleitet. (Nicht-JSON-Nachrichten werden an eine von Ihnen konfigurierte `$default`-Route weitergeleitet.)

Eine Route umfasst einen Routenschlüssel. Dabei handelt es sich um den Wert, der bei der Auswertung eines Routen-Auswahlausdrucks erwartet wird. Das `routeSelectionExpression` ist ein auf API-Ebene definiertes Attribut. Es gibt eine JSON-Eigenschaft an, von der erwartet wird, dass sie in der Nachricht-Nutzlast vorhanden ist. Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter [the section called “”](#).

Beispiel: Wenn Ihre JSON-Nachrichten eine `action`-Eigenschaft enthalten und Sie basierend auf dieser Eigenschaft verschiedene Aktionen durchführen möchten, könnte Ihr Routen-Auswahlausdruck `${request.body.action}` lauten. Ihre Routing-Tabelle würde in diesem Fall angeben, welche Aktion ausgeführt werden soll, indem der Wert der `action`-Eigenschaft gegen die benutzerdefinierten Routenschlüssel-Werte abgeglichen wird, die Sie in der Tabelle definiert haben.

Verwenden Sie Routen für eine API WebSocket

Es gibt drei vordefinierte Routen, die verwendet werden können: `$connect`, `$disconnect` und `$default`. Darüber hinaus können Sie benutzerdefinierte Routen erstellen.

- API Gateway ruft die `$connect` Route auf, wenn eine persistente Verbindung zwischen dem Client und einer WebSocket API initiiert wird.
- API Gateway ruft die `$disconnect`-Route auf, wenn der Client oder der Server die Verbindung mit der API unterbricht.
- API Gateway ruft eine benutzerdefinierte Route auf, wenn nach der Auswertung des Routen-Auswahlausdrucks im Hinblick auf die Nachricht eine übereinstimmende Route gefunden wird. Die Übereinstimmung bestimmt, welche Integration aufgerufen wird.
- API Gateway ruft die Route `$default` auf, wenn der Routen-Auswahlausdruck nicht im Hinblick auf die Nachricht ausgewertet werden kann oder wenn keine übereinstimmende Route gefunden wird.

Weitere Informationen über die Routen `$connect` und `$disconnect` finden Sie unter [the section called “Verwalten von verbundenen Benutzern und Client-Apps”](#).

Weitere Informationen über die Route `$default` und benutzerdefinierte Routen finden Sie unter [the section called “Aufrufen Ihrer Backend-Integration”](#).

Daten an verbundene Client-Apps senden

Backend-Services können Daten an verbundene Client-Apps senden. Gehen Sie für eine Datenversendung wie folgt vor:

- Senden Sie eine Antwort mithilfe einer Integration. Diese wird über die von Ihnen definierte Routenantwort an den Client zurückgegeben.
- Sie können mithilfe der `@connections`-API eine POST-Anforderung senden. Weitere Informationen finden Sie unter [the section called “Verwenden der @connections-Befehle in Ihrem Backend-Service”](#).

WebSocket API-Statuscodes

API Gateway WebSocket APIs verwendet die folgenden Statuscodes für die Kommunikation vom Server zum Client, wie in der [WebSocket Close Code Number Registry](#) beschrieben:

1001

API Gateway gibt diesen Statuscode zurück, wenn der Client 10 Minuten inaktiv ist oder die maximale Verbindungsdauer von 2 Stunden erreicht.

1003

API Gateway gibt diesen Statuscode zurück, wenn ein Endpunkt einen binären Medientyp empfängt. Binäre Medientypen werden nicht unterstützt für WebSocket APIs.

1005

API Gateway gibt diesen Statuscode zurück, wenn der Client ein Close-Frame ohne Abschlusscode sendet.

1006

API Gateway gibt diesen Statuscode zurück, wenn die Verbindung unerwartet geschlossen wird, z. B. wenn die TCP-Verbindung ohne einen geschlossenen Frame WebSocket geschlossen wurde.

1008

API Gateway gibt diesen Statuscode zurück, wenn ein Endpunkt zu viele Anfragen von einem bestimmten Client erhält.

1009

API Gateway gibt diesen Statuscode zurück, wenn ein Endpunkt eine Nachricht empfängt, die zu groß ist, um verarbeitet zu werden.

1011

API Gateway gibt diesen Statuscode zurück, wenn ein interner Serverfehler auftritt.

1012

API Gateway gibt diesen Statuscode zurück, wenn der Service neu gestartet wird.

Verwalten von verbundenen Benutzern und Client-Apps: **\$connect**- und **\$disconnect**-Routen

Im folgenden Abschnitt wird beschrieben, wie Sie die `$connect` und `$disconnect` -Routen für Ihre WebSocket API verwenden.

Themen

- [Die Route `\$connect`](#)
- [Verbindungsinformationen von der `\$connect`-Route übergeben](#)
- [Die Route `\$disconnect`](#)

Die Route **\$connect**

Client-Apps stellen eine Verbindung zu Ihrer WebSocket API her, indem sie eine WebSocket Upgrade-Anfrage senden. Wenn die Anforderung erfolgreich ist, wird die Route `$connect` ausgeführt, während die Verbindung hergestellt wird.

Da es sich bei der WebSocket Verbindung um eine statusbehaftete Verbindung handelt, können Sie die Autorisierung nur für die `$connect` Route konfigurieren. AuthN/AuthZ wird nur zur Verbindungszeit ausgeführt.

Solange die Integration im Zusammenhang mit der Route `$connect` noch nicht abgeschlossen wurde, steht die Upgrade-Anforderung noch an und die tatsächliche Verbindung wird nicht hergestellt. Wenn die `$connect`-Anforderung fehlschlägt (z. B. aufgrund eines AuthN/AuthZ-Fehlers oder eines Integrationsfehlers), wird die Verbindung nicht hergestellt.

 Note

Wenn die Autorisierung für `$connect` fehlschlägt, wird keine Verbindung hergestellt, und der Client erhält eine 401- oder 403-Antwort.

Das Einrichten einer Integration für `$connect` ist optional. Sie sollten die Einrichtung einer `$connect`-Integration unter folgenden Umständen erwägen:

- Sie möchten Clients ermöglichen, Unterprotokolle mithilfe des `Sec-WebSocket-Protocol`-Felds anzugeben. Beispielcode finden Sie unter [Richten Sie eine `\$connect` Route ein, für die ein WebSocket Unterprotokoll erforderlich ist](#).
- Sie möchten benachrichtigt werden, wenn Clients verbunden werden.
- Wenn Sie Verbindungen drosseln möchten oder wenn Sie steuern möchten, wer eine Verbindung herstellt.
- Wenn Sie möchten, dass Ihr Backend über eine Rückruf-URL Nachrichten an Clients zurücksendet.
- Sie möchten jede Verbindungs-ID und andere Informationen in einer Datenbank (z. B. Amazon DynamoDB) speichern.

Verbindungsinformationen von der `$connect`-Route übergeben

Sie können sowohl Proxy- als auch Nicht-Proxy-Integrationen verwenden, um Informationen von der `$connect`-Route an eine Datenbank oder einen anderen AWS-Service zu übergeben.

So übergeben Sie Verbindungsinformationen mithilfe einer Proxy-Integration

In diesem Fall können Sie über eine Lambda-Proxy-Integration auf die Verbindungsinformationen zugreifen. Verwenden Sie eine andere AWS-Service AWS Lambda OR-Funktion, um Beiträge an die Verbindung zu senden.

Die folgende Lambda-Funktion zeigt, wie das `requestContext`-Objekt verwendet wird, um die Verbindungs-ID, den Domännennamen, den Stufennamen und die Abfragezeichenfolgen zu protokollieren.

Node.js

```
export const handler = async(event, context) => {
```

```
const connectId = event["requestContext"]["connectionId"]
const domainName = event["requestContext"]["domainName"]
const stageName = event["requestContext"]["stage"]
const qs = event['queryStringParameters']
console.log('Connection ID: ', connectId, 'Domain Name: ', domainName, 'Stage
Name: ', stageName, 'Query Strings: ', qs )
return {"statusCode" : 200}
};
```

Python

```
import json
import logging
logger = logging.getLogger()
logger.setLevel("INFO")

def lambda_handler(event, context):
    connectId = event["requestContext"]["connectionId"]
    domainName = event["requestContext"]["domainName"]
    stageName = event["requestContext"]["stage"]
    qs = event['queryStringParameters']
    connectionInfo = {
        'Connection ID': connectId,
        'Domain Name': domainName,
        'Stage Name': stageName,
        'Query Strings': qs}
    logging.info(connectionInfo)
    return {"statusCode": 200}
```

So übergeben Sie Verbindungsinformationen mithilfe einer Nicht-Proxy-Integration

- Sie können mit einer Nicht-Proxy-Integration auf die Verbindungsinformationen zugreifen. Richten Sie die Integrationsanfrage ein und stellen Sie eine WebSocket API-Anforderungsvorlage bereit. Die folgende Zuordnungsvorlage der [Velocity Template Language \(VTL\)](#) enthält eine Integrationsanforderung. Diese Anforderung sendet die folgenden Details an eine Nicht-Proxy-Integration:

- Verbindungs-ID
- Domainname

- Stufenname
- Pfad
- Kopfzeilen
- Abfragezeichenfolgen

Diese Anforderung sendet die Verbindungs-ID, den Domain-Namen, den Namen der Phase, die Pfade, Header und Abfragezeichenfolgen an eine Nicht-Proxy-Integration.

```
{
  "connectionId": "$context.connectionId",
  "domain": "$context.domainName",
  "stage": "$context.stage",
  "params": "$input.params()"
}
```

Weitere Informationen zum Einrichten von Datenübertragungen finden Sie unter [the section called “Datentransformationen”](#).

Zum Abschließen der Integrationsanforderung legen Sie die Integrationsantwort `Status Code: 200` fest. Weitere Informationen zum Einrichten einer Integrationsantwort finden Sie unter [Integrationsantwort mit der API Gateway-Konsole einrichten](#).

Die Route **\$disconnect**

Die Route `$disconnect` wird ausgeführt, nachdem die Verbindung geschlossen wurde.

Die Verbindung kann vom Server oder vom Client geschlossen werden. Da die Verbindung bereits geschlossen ist, wenn sie ausgeführt wird, ist `$disconnect` ein "Best Effort"-Ereignis. API Gateway wird sich nach bemühen, das `$disconnect`-Ereignis an Ihre Integration zu übermitteln, kann aber die Übermittlung nicht garantieren.

Das Backend kann die Verbindungstrennung mithilfe der `@connections`-API einleiten. Weitere Informationen finden Sie unter [the section called “Verwenden der @connections-Befehle in Ihrem Backend-Service”](#).

Aufrufen Ihrer Backend-Integration mit der `$default`-Route und benutzerdefinierten Routen in API Gateway

Im folgenden Abschnitt wird beschrieben, wie Sie Ihre Backend-Integration entweder mithilfe der `$default` Route oder einer benutzerdefinierten Route für eine WebSocket API aufrufen.

Themen

- [Verwenden von Routen zur Verarbeitung von Nachrichten](#)
- [Die Route `\$default`](#)
- [Benutzerdefinierte Routen](#)
- [Verwenden Sie API Gateway WebSocket API-Integrationen, um eine Verbindung zu Ihrer Geschäftslogik herzustellen](#)
- [Wichtige Unterschiede zwischen WebSocket APIs und REST APIs](#)

Verwenden von Routen zur Verarbeitung von Nachrichten

In API Gateway WebSocket APIs können Nachrichten vom Client an Ihren Backend-Service gesendet werden und umgekehrt. Im Gegensatz zum request/response HTTP-Modell können im Backend Nachrichten an WebSocket den Client gesendet werden, ohne dass der Client etwas unternimmt.

Es kann sich dabei um JSON- oder Nicht-JSON-Nachrichten handeln. Es können jedoch nur JSON-Nachrichten basierend auf dem Nachrichteninhalt an bestimmte Integrationen weitergeleitet werden. Nicht-JSON-Nachrichten werden über die `$default`-Route an das Backend übergeben.

Note

API Gateway unterstützt Nachrichten-Payloads bis zu 128 KB mit einer maximalen Frame-Größe von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind. Wenn eine größere Nachricht (oder ein größerer Frame) empfangen wird, wird die Verbindung mit dem Code 1009 geschlossen.

Derzeit werden keine binären Nutzlasten unterstützt. Wenn ein binärer Frame empfangen wird, wird die Verbindung mit dem Code 1003 geschlossen. Es ist jedoch möglich, binäre Nutzlasten in Text zu konvertieren. Siehe [the section called “Binäre Medientypen”](#).

WebSocket APIs In API Gateway können JSON-Nachrichten weitergeleitet werden, um einen bestimmten Backend-Service basierend auf dem Nachrichteninhalt auszuführen. Wenn ein Client eine Nachricht über seine WebSocket Verbindung sendet, führt dies zu einer Routing-Anfrage an die WebSocket API. Die Anfrage wird der Route mit dem entsprechenden Routenschlüssel in API Gateway zugeordnet. Sie können eine Routenanforderung für eine WebSocket API in der API Gateway Gateway-Konsole einrichten AWS CLI, indem Sie das oder ein AWS SDK verwenden.

Note

Im AWS CLI und können Sie Routen erstellen AWS SDKs, bevor oder nachdem Sie Integrationen erstellt haben. Derzeit unterstützt die Konsole keine Wiederverwendung von Integrationen. Daher müssen Sie zuerst die Route erstellen und dann die Integration für diese Route.

Sie können API Gateway so konfigurieren, dass die Validierung einer Routenanforderung durchgeführt wird, bevor Sie mit der Integrationsanforderung fortfahren. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage fehl, ohne Ihr Backend aufzurufen, sendet eine "Bad request body" Gateway-Antwort ähnlich der folgenden an den Client und veröffentlicht die Validierungsergebnisse in CloudWatch Logs:

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId": "{messageId}"}
```

Damit werden unnötige Aufrufe Ihres Backend reduziert, sodass Sie sich auf andere Voraussetzungen für Ihre API konzentrieren können.

Darüber hinaus können Sie eine Routenantwort für die Routen Ihrer API vorgeben, um eine bidirektionale Kommunikation zu ermöglichen. Eine Routenantwort beschreibt, welche Daten bei Abschluss der Integration einer bestimmten Route an Ihren Client gesendet werden. Es ist nicht notwendig, eine Antwort für eine Route vorzugeben, wenn ein Client Nachrichten an Ihr Backend senden soll, ohne eine Antwort zu erhalten (unidirektionale Kommunikation). Wenn Sie keine Routenantwort bereitstellen, sendet API Gateway jedoch keine Informationen über das Ergebnis Ihrer Integration an Ihre Kunden.

Die Route **\$default**

Jede WebSocket API-Gateway-API kann eine `$default` Route haben. Hierbei handelt es sich um einen speziellen Routing-Wert, der auf folgende Weise eingesetzt werden kann:

- Sie können ihn zusammen mit vorgegebenen Routenschlüsseln verwenden, um für eingehende Nachrichten, auf die keine der vorgegebenen Routenschlüssel zutreffen, eine "Fallback"-Route anzugeben (z. B. eine generische Pseudo-Integration, die eine bestimmte Fehlermeldung zurückgibt).
- Sie können ihn ohne vorgegebene Routenschlüssel verwenden, um ein Proxy-Modell anzugeben, durch das Routing an eine Backend-Komponente delegiert wird.
- Sie können mit ihm eine Route für Nicht-JSON-Nutzlasten angeben.

Benutzerdefinierte Routen

Wenn basierend auf dem Nachrichteninhalt eine bestimmte Integration aufgerufen werden soll, können Sie hierzu eine benutzerdefinierte Route erstellen.

Für eine benutzerdefinierte Route werden ein Routenschlüssel und eine Integration nach Ihren Angaben verwendet. Wenn eine eingehende Nachricht eine JSON-Eigenschaft enthält und diese Eigenschaft mit einem Wert ausgewertet wird, der mit dem Routenschlüsselwert übereinstimmt, ruft API Gateway die Integration auf. (Weitere Informationen finden Sie unter [the section called “Überblick über WebSocket APIs”](#).)

Angenommen, Sie möchten eine Chat-Raum-Anwendung erstellen. Sie könnten damit beginnen, eine WebSocket API zu erstellen, deren Routenauswahlausdruck lautet `$request.body.action`. Anschließend könnten Sie zwei Routen definieren: `joinroom` und `sendmessage`. Ein Client-App könnte die `joinroom`-Route durch Senden einer Nachricht wie die folgende aufrufen:

```
{"action":"joinroom","roomname":"developers"}
```

Und sie könnte die `sendmessage`-Route durch Senden einer Nachricht wie die folgende aufrufen:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Verwenden Sie API Gateway WebSocket API-Integrationen, um eine Verbindung zu Ihrer Geschäftslogik herzustellen

Nachdem Sie eine Route für eine WebSocket API-Gateway-API eingerichtet haben, müssen Sie die Integration angeben, die Sie verwenden möchten. Genauso wie eine Route über eine Routenanforderung und eine Routenantwort verfügen kann, kann eine Integration eine Integrationsanforderung und eine Integrationsantwort besitzen. Eine Integrationsanforderung enthält

die Informationen, die von Ihrem Backend zur Verarbeitung der von Ihrem Client stammenden Anforderung erwartet werden. Die Integrationsantwort enthält die Daten, die Ihr Backend an API Gateway zurücksendet. Sie kann verwendet werden, um eine an den Client zu sendende Nachricht zu verfassen (falls eine Routenantwort definiert ist).

Weitere Informationen zum Einrichten von Integrationen finden Sie unter [the section called "Integrationen"](#).

Wichtige Unterschiede zwischen WebSocket APIs und REST APIs

Integrationen für WebSocket APIs ähneln Integrationen für REST APIs, mit Ausnahme der folgenden Unterschiede:

- Derzeit müssen Sie in der API Gateway-Konsole zuerst eine Route und dann eine Integration als Ziel dieser Route erstellen. In der API und der CLI können Sie Routen und Integrationen jedoch unabhängig voneinander in beliebiger Reihenfolge erstellen.
- Sie können eine einzelne Integration für mehrere Routen verwenden. Für eine Gruppe von Aktionen, die eng miteinander in Beziehung stehen, empfiehlt es sich beispielsweise, dass alle diese Routen zu einer einzelnen Lambda-Funktion führen. Anstatt die Details der Integration mehrmals zu definieren, können Sie sie einmal angeben und jeder verwandten Route zuweisen.

Note

Derzeit unterstützt die Konsole keine Wiederverwendung von Integrationen. Daher müssen Sie zuerst die Route erstellen und dann die Integration für diese Route.

Im Feld AWS CLI und können Sie eine Integration wiederverwenden AWS SDKs, indem Sie das Ziel der Route auf einen Wert von setzen `"integrations/{integration-id}"`, wobei `{integration-id}` es sich um die eindeutige ID der Integration handelt, die der Route zugeordnet werden soll.

- API Gateway bietet mehrere [Auswahlausdrücke](#), die Sie in Ihren Routen und Integrationen verwenden können. Sie müssen die Auswahl einer Eingabevorlage oder einer Ausgabezuordnung nicht vom Inhaltstyp abhängig machen. Wie bei Routenauswahlausdrücken können Sie einen Auswahlausdruck definieren, der von API Gateway ausgewertet wird, um das richtige Element auszuwählen. Alle von ihnen übernehmen automatisch die `$default`-Vorlage, wenn keine passende Vorlage gefunden wird.
- In Integrationsanforderungen unterstützt der Vorlagen-Auswahlausdruck `$request.body.<json_path_expression>` und statische Werte.

- In Integrationsantworten unterstützt der Vorlagen-Auswahlausdruck `$request.body.<json_path_expression>`, `$integration.response.statuscode` und `$integration.response.header.<headerName>` sowie statische Werte.

Im HTTP-Protokoll, in dem Anforderungen und Antworten synchron gesendet werden, ist die Kommunikation im Wesentlichen unidirektional. Im WebSocket Protokoll erfolgt die Kommunikation in beide Richtungen. Antworten sind asynchron und werden vom Client nicht unbedingt in der gleichen Reihenfolge empfangen, in der die Client-Nachrichten gesendet wurden. Darüber hinaus kann das Backend Nachrichten an den Client senden.

Note

Für eine Route, die zur Verwendung der Integration `AWS_PROXY` oder `LAMBDA_PROXY` konfiguriert wurde, ist die Kommunikation unidirektional. API Gateway übergibt die Backend-Antwort nicht automatisch über die Routenantwort. Wenn bei der `LAMBDA_PROXY`-Integration wird der von der Lambda-Funktion zurückgegebene Text beispielsweise nicht an den Client zurückgegeben. Wenn Sie möchten, dass der Client Integrationsantworten erhält, müssen Sie eine Routenantwort definieren, um eine bidirektionale Kommunikation zu ermöglichen.

WebSocket Auswahlausdrücke

API Gateway verwendet Auswahlausdrücke als Möglichkeit, den Anfrage- und Antwortkontext auszuwerten und einen Schlüssel zu erzeugen. Anhand des Schlüssels wird dann aus einem Satz möglicher Werte ausgewählt, die gewöhnlich von Ihnen, dem API-Entwickler, bereitgestellt werden. Der genaue Satz der unterstützten Variablen ist vom jeweiligen Ausdruck abhängig. Jeder Ausdruck wird nachstehend weiter erörtert.

Für alle Ausdrücke folgt die Sprache dem gleichen Satz von Regeln:

- Einer Variable wird vorangestellt "\$".
- Variablen können explizit durch geschweifte Klammern abgegrenzt werden, z. ., `"${request.body.version}-beta"`.
- Mehrere Variablen werden unterstützt, aber die Auswertung findet nur einmal statt (keine rekursive Auswertung).

- Ein Dollarzeichen (\$) kann durch die Escape-Zeichen "\\" geschützt werden. Dies ist besonders nützlich, wenn ein Ausdruck definiert wird, der dem reservierten `$default`-Schlüssel zugeordnet wird, z. B. "\\\$default".
- In einigen Fällen ist ein Musterformat erforderlich. In diesem Fall sollte der Ausdruck entsprechend `"/"`-Statuscodes mit Schrägstrichen (`"/2\d\d/"`) verpackt werden, z. B. `2XX`.

Themen

- [Routenantwort-Auswahlausdrücke](#)
- [API-Schlüssel-Auswahlausdrücke](#)
- [API-Zuordnungs-Auswahlausdrücke](#)
- [WebSocket Zusammenfassung des Auswahlausdrucks](#)

Routenantwort-Auswahlausdrücke

Eine [Routenantwort](#) wird für die Modellierung einer Antwort vom Backend zum Client verwendet. Für WebSocket APIs ist eine Routenantwort optional. Wenn es definiert ist, signalisiert es API Gateway, dass es beim Empfang einer WebSocket Nachricht eine Antwort an einen Client zurückgeben soll.

Die Auswertung des Routenantwort-Auswahlausdrucks ergibt einen Routenantwort-Schlüssel. Zukünftig wird dieser Schlüssel zur Auswahl unter einer der mit der API verknüpften [RouteResponses](#) dienen. Zurzeit wird jedoch nur der Schlüssel `$default` unterstützt.

API-Schlüssel-Auswahlausdrücke

Dieser Ausdruck wird ausgewertet, wenn der Service bestimmt, dass mit der vorliegenden Anforderung nur fortgefahren werden soll, wenn der Client einen gültigen [API-Schlüssel](#) bereitstellt.

Zurzeit werden als einzigen Werte `$request.header.x-api-key` und `$context.authorizer.usageIdentifierKey` unterstützt.

API-Zuordnungs-Auswahlausdrücke

Dieser Ausdruck wird ausgewertet, um zu bestimmen, welche API-Stufe bei einer über eine benutzerdefinierte Domäne gestellten Anforderung auszuwählen ist.

Derzeit wird als einziger Wert unterstützt `$request.basepath`.

WebSocket Zusammenfassung des Auswahlausdrucks

In der folgenden Tabelle sind die Anwendungsfälle für Auswahlausdrücke zusammengefasst in WebSocket APIs:

Auswahlau sdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispiela nwendungs fall
<code>Api.Route Selection Expression</code>	<code>Route.RouteKey</code>	<code>\$default</code> wird als Catch-all-Route unterstützt.	Leitet WebSocket Nachrichten auf der Grundlage des Kontextes einer Client-Anfrage weiter.
<code>Route.ModelSelecti onExpress ion</code>	Schlüssel für <code>Route.RequestModels</code>	Optional. Bei Angabe für Nicht-Proxy-Integrationen findet eine Modellvalidierung statt.	Führt die Anforderungsvalidierung dynamisch innerhalb derselben Route aus.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
		<code>\$default</code> wird als Catch-all-Route unterstützt.	

Auswahlau sdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispiela nwendungs fall
Integrati on.Templa teSelecti onExpress ion	Schlüssel für Integration.RequestTemplates	Optional. Kann für Nicht- Proxy- Integr ationen zum Bearbeite n von eingeher en Nutzlaste n bereitges tellt werden. `\${requer .body. ; nPath} zte Werte sind und statische Werte. \$default wird als Catch-	Dient zur Bearbeitu ng der Anforderu ng der aufrufend en Methode je nach den dynamisch en Eigenscha ften der Anforderu ng.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispiela nwendungs fall
		all- Route unterstüt- zt.	

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungen
<code>IntegrationResponse.SelectionExpression</code>	<code>IntegrationResponse.SelectionKey</code>	Optional. Kann für Nicht-Proxy-Integrationen bereitgestellt werden. Dient als Mustervergleich für Fehlermeldungen (von Lambda oder Statuscodes (von HTTP-Integrationen)). \$default ist erforderlich, wenn	Dient zum Bearbeiten der Antwort vom Backend. Wählen Sie, welche Aktion basierend auf der dynamischen Antwort des Backends ausgeführt werden soll (z. B. unterschiedliche Behandlungen bestimmter Fehler).

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
		damit Nicht-Proxy-Integrationen als Catch-All für erfolgreiche Antworten fungieren.	

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
IntegrationResponse.TemplateSelectionExpression	Schlüssel für IntegrationResponse.ResponseTemplates	Optional. Kann für Nicht-Proxy-Integrationen bereitgestellt werden. \$default wird unterstützt.	In einigen Fällen kann eine dynamische Eigenschaft der Antwort verschiedene Transformationen innerhalb derselben Route und zugehörigen Integration vorgeben. <pre> \${request .body.jsonPath} , \${integration.response.statusCode} , \${integration.resp </pre>

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
			<code>response.header.headerName</code> , <code>\${integration.response.multiHeaderValue.header.headerName}</code> , Unterstützte Werte sind , , , und statische Werte. <code>\$default</code> wird als Catch-all-Route unterstützt.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
<code>Route.RouteResponseSelectionExpression</code>	<code>RouteResponse.RouteResponseKey</code>	Sollte bereitgestellt werden, um die bidirektionale Kommunikation für eine WebSocket Route zu initiieren. Zurzeit ist dieser Wert ausschließlich auf <code>\$default</code> eingeschränkt.	
<code>RouteResponse.ModelSelectionExpression</code>	Schlüssel für <code>RouteResponse.RequestModels</code>	Wird derzeit nicht unterstützt.	

WebSocket APIs In API Gateway entwickeln

Dieser Abschnitt enthält Einzelheiten zu den API Gateway Gateway-Funktionen, die Sie bei der Entwicklung Ihres API Gateway benötigen APIs.

Während Sie Ihre API Gateway-API entwickeln, entscheiden Sie sich für eine Reihe von Merkmalen Ihrer API. Diese Eigenschaften hängen davon ab, wofür Ihre API verwendet werden soll. So könnte es beispielsweise sein, dass Sie es nur bestimmten Clients gestatten möchten, die API aufzurufen. Vielleicht soll die API aber auch für alle verfügbar sein. Vielleicht benötigen Sie einen API-Aufruf, um eine Lambda-Funktion auszuführen, eine Datenbankabfrage durchzuführen oder eine Anwendung aufzurufen.

Themen

- [WebSocket APIs Im API Gateway erstellen](#)
- [IP-Adresstypen für WebSocket-APIs im API Gateway](#)
- [Routen für WebSocket APIs im API Gateway erstellen](#)
- [Steuern und verwalten Sie den Zugriff WebSocket APIs auf das API Gateway](#)
- [Integrationen für das WebSocket APIs API Gateway](#)
- [Validierung für WebSocket APIs im API Gateway anfordern](#)
- [Datentransformationen für WebSocket APIs im API Gateway](#)
- [Binäre Medientypen für WebSocket-APIs in API Gateway](#)
- [Aufrufen WebSocket APIs](#)

WebSocket APIs Im API Gateway erstellen

Sie können eine WebSocket API in der API Gateway Gateway-Konsole erstellen, indem Sie den Befehl AWS CLI [create-api](#) oder den `CreateApi` Befehl in einem AWS SDK verwenden. Die folgenden Verfahren zeigen, wie Sie eine neue WebSocket API erstellen.

Note

WebSocket APIs unterstützt nur TLS 1.2 und TLS 1.3. Frühere TLS-Versionen werden nicht unterstützt.

Erstellen Sie eine WebSocket API mithilfe von AWS CLI Befehlen

Der folgende [create-api](#)-Befehl erstellt eine API mit dem Routen-Auswahlausdruck `$request.body.action`:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type WEBSOCKET --route-selection-expression '$request.body.action'
```

Die Ausgabe sollte wie folgt aussehen:

```
{
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "Name": "myWebSocketApi3",
  "CreateDate": "2018-11-15T06:23:51Z",
  "ProtocolType": "WEBSOCKET",
  "RouteSelectionExpression": "'$request.body.action'",
  "ApiId": "aabbccdde"
}
```

Erstellen Sie eine WebSocket API mit der API Gateway Gateway-Konsole

Sie können eine WebSocket API in der Konsole erstellen, indem Sie das WebSocket Protokoll auswählen und der API einen Namen geben.

Important

Sobald Sie die API erstellt haben, können Sie das für sie ausgewählte Protokoll nicht mehr ändern. Es gibt keine Möglichkeit, eine WebSocket API in eine REST-API umzuwandeln oder umgekehrt.

So erstellen Sie eine WebSocket API mit der API Gateway Gateway-Konsole

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie Create API (API erstellen).
2. Wählen Sie unter WebSocket API die Option Build aus. Es werden nur regionale Endpunkte unterstützt.
3. Geben Sie unter API-Name den Namen Ihrer API ein.
4. Geben Sie unter Ausdruck für die Routenauswahl einen Wert ein. Beispiel, `$request.body.action`.

Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter [the section called “](#)

5. Führen Sie eine der folgenden Aktionen aus:

- Wählen Sie Leere API erstellen aus, wenn Sie eine API ohne Routen erstellen möchten.
- Klicken Sie auf Weiter, um Ihrer API Routen anzuhängen.

Sie können Routen anhängen, nachdem Sie Ihre API erstellt haben.

IP-Adresstypen für WebSocket-APIs im API Gateway

Beim Erstellen einer API geben Sie den Typ der IP-Adressen an, die Ihre API aufrufen dürfen. Sie können IPv4 auswählen, um IPv4-Adressen für den Aufruf Ihrer API zuzulassen, oder Sie können Dualstack auswählen, um sowohl IPv4- als auch IPv6-Adressen zuzulassen. Wir empfehlen, den IP-Adresstyp auf Dualstack zu setzen, um Engpässe im IP-Adressraum zu vermeiden und Ihre Sicherheitsposition zu verbessern. Weitere Informationen zu den Vorteilen eines Dualstack-IP-Adresstyps finden Sie unter [IPv6 in AWS](#).

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können Ihre Verwendung von IP-Adresstypen beeinflussen:

- Der Standard-IP-Adresstyp für alle WebSocket-APIs ist IPv4.
- Wenn Sie den IP-Adresstyp einer bestehenden API von IPv4 in Dualstack ändern, stellen Sie sicher, dass alle Richtlinien, die den Zugriff auf Ihre APIs steuern, aktualisiert wurden, um IPv6-Aufrufe zu berücksichtigen. Wenn Sie den IP-Adresstyp ändern, wird die Änderung sofort wirksam.
- Ihre API kann einem benutzerdefinierten Domainnamen mit einem anderen IP-Adresstyp als Ihre API zugewiesen werden. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.

Ändern des IP-Adresstyps einer WebSocket-API

Sie können den IP-Adresstyp ändern, indem Sie die API-Konfiguration aktualisieren. Dies ist über die AWS-Managementkonsole, die AWS CLI, CloudFormation oder ein AWS SDK möglich. Wenn Sie den IP-Adresstyp der API ändern, müssen Sie Ihre API nicht erneut bereitstellen, damit die Änderungen wirksam werden.

AWS-Managementkonsole

So ändern Sie den IP-Adresstypen einer WebSocket-API

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine WebSocket-API.
3. Wählen Sie API-Einstellungen und anschließend Bearbeiten aus.
4. Wählen Sie unter „IP-Adresstyp“ entweder IPv4 oder Dualstack aus.
5. Wählen Sie Speichern aus.

Die Änderung der API-Konfiguration wird sofort wirksam.

AWS CLI

Der folgende [update-api](#)-Befehl aktualisiert eine API, sodass sie den IP-Adresstyp Dualstack verwendet:

```
aws apigatewayv2 update-api \  
  --api-id abcd1234 \  
  --ip-address-type dualstack
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",  
  "ApiId": "abcd1234",  
  "ApiKeySelectionExpression": "$request.header.x-api-key",  
  "CreateDate": "2025-02-04T22:20:20+00:00",  
  "DisableExecuteApiEndpoint": false,  
  "Name": "My-WebSocket-API",  
  "ProtocolType": "WEBSOCKET",  
  "RouteSelectionExpression": "$request.method $request.path",  
  "Tags": {},  
  "NotificationUri": [],  
  "IpAddressType": "dualstack"  
}
```

Routen für WebSocket APIs im API Gateway erstellen

In Ihrer WebSocket API werden eingehende JSON-Nachrichten auf der Grundlage von Routen, die Sie konfigurieren, an Backend-Integrationen weitergeleitet. (Nicht-JSON-Nachrichten werden an eine von Ihnen konfigurierte `$default`-Route weitergeleitet.)

Eine Route umfasst einen Routenschlüssel. Dabei handelt es sich um den Wert, der bei der Auswertung eines Routen-Auswahlausdrucks erwartet wird. Das `routeSelectionExpression` ist ein auf API-Ebene definiertes Attribut. Es gibt eine JSON-Eigenschaft an, von der erwartet wird, dass sie in der Nachricht-Nutzlast vorhanden ist. Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter [the section called “”](#).

Beispiel: Wenn Ihre JSON-Nachrichten eine `action`-Eigenschaft enthalten und Sie basierend auf dieser Eigenschaft verschiedene Aktionen durchführen möchten, könnte Ihr Routen-Auswahlausdruck `${request.body.action}` lauten. Ihre Routing-Tabelle würde in diesem Fall angeben, welche Aktion ausgeführt werden soll, indem der Wert der `action`-Eigenschaft gegen die benutzerdefinierten Routenschlüssel-Werte abgeglichen wird, die Sie in der Tabelle definiert haben.

Es gibt drei vordefinierte Routen, die verwendet werden können: `$connect`, `$disconnect` und `$default`. Darüber hinaus können Sie benutzerdefinierte Routen erstellen.

- API Gateway ruft die `$connect` Route auf, wenn eine persistente Verbindung zwischen dem Client und einer WebSocket API initiiert wird.
- API Gateway ruft die `$disconnect`-Route auf, wenn der Client oder der Server die Verbindung mit der API unterbricht.
- API Gateway ruft eine benutzerdefinierte Route auf, wenn nach der Auswertung des Routen-Auswahlausdrucks im Hinblick auf die Nachricht eine übereinstimmende Route gefunden wird. Die Übereinstimmung bestimmt, welche Integration aufgerufen wird.
- API Gateway ruft die Route `$default` auf, wenn der Routen-Auswahlausdruck nicht im Hinblick auf die Nachricht ausgewertet werden kann oder wenn keine übereinstimmende Route gefunden wird.

Routen-Auswahlausdrücke

Ein Routen-Auswahlausdruck wird ausgewertet, wenn der Service die Route auswählt, der eine eingehende Nachricht folgen soll. Der Service verwendet die Route, deren `routeKey` genau dem ausgewerteten Wert entspricht. Wenn bei fehlender Übereinstimmung eine Route

mit dem Routenschlüssel `$default` vorhanden ist, wird diese Route ausgewählt. Wenn keine Routen dem ausgewählten Wert entsprechen und keine `$default`-Route vorhanden ist, gibt der Service einen Fehler zurück. Für WebSocket -based APIs sollte der Ausdruck die folgende Form haben `$request.body.{path_to_body_element}`.

Angenommen, Sie senden die folgende JSON-Nachricht:

```
{
  "service" : "chat",
  "action" : "join",
  "data" : {
    "room" : "room1234"
  }
}
```

Möglicherweise möchten Sie das Verhalten Ihrer API basierend auf der Eigenschaft `action` auswählen. In diesem Fall könnten Sie den folgenden Routen-Auswahlausdruck definieren:

```
$request.body.action
```

`request.body` Bezieht sich in diesem Beispiel auf die JSON-Nutzlast Ihrer Nachricht und `.action` ist ein [JSONPath](#) Ausdruck. Sie können nach `request.body` einen beliebigen JSON-Pfad-Ausdruck verwenden. Denken Sie aber daran, dass das Ergebnis in Text umgewandelt wird. Wenn Ihr JSONPath Ausdruck beispielsweise ein Array mit zwei Elementen zurückgibt, wird dieses als Zeichenfolge `"[item1, item2]"` dargestellt. Aus diesem Grund ist es sinnvoll, dass Ihr Ausdruck auf einen Wert und nicht auf ein Array oder Objekt ausgewertet wird.

Sie können einfach einen statischen Wert oder auch mehrere Variablen verwenden. In der folgenden Tabelle werden Beispiele und deren im Hinblick auf die vorhergehende Nutzlast ausgewerteten Ergebnisse gezeigt.

Ausdruck	Ausgewertetes Ergebnis	Beschreibung
<code>\$request.body.action</code>	<code>join</code>	Eine entpackte Variable

Ausdruck	Ausgewertetes Ergebnis	Beschreibung
<code>\${request.body.action}</code>	<code>join</code>	Eine verpackte Variable
<code>\${request.body.service}/\${request.body.action}</code>	<code>chat/join</code>	Mehrere Variablen mit statischen Werten
<code>\${request.body.action}-\${request.body.invalidPath}</code>	<code>join-</code>	Wenn der nicht gefundene JSONPath wird, wird die Variable als „.“ aufgelöst.
<code>action</code>	<code>action</code>	Statischer Wert
<code>\\$default</code>	<code>\$default</code>	Statischer Wert

Das ausgewertete Ergebnis wird zum Auffinden der Route verwendet. Wenn eine Route mit einem übereinstimmenden Routenschlüssel vorhanden ist, wird die Route zur Verarbeitung der Nachricht ausgewählt. Wenn keine passende Route gefunden wird, versucht API Gateway, die `$default`-Route zu finden, falls verfügbar. Wenn die `$default`-Route nicht definiert ist, gibt API Gateway einen Fehler zurück.

Routen für eine WebSocket API in API Gateway einrichten

Wenn Sie zum ersten Mal eine neue WebSocket API erstellen, gibt es drei vordefinierte Routen: `$connect`, `$disconnect`, und `$default`. Sie können sie mithilfe der Konsole, der API oder erstellen AWS CLI. Auf Wunsch können Sie benutzerdefinierte Routen erstellen. Weitere Informationen finden Sie unter [the section called “Überblick über WebSocket APIs”](#).

Note

In der CLI ist es möglich, Routen vor oder nach dem Erstellen von Integrationen zu erstellen. Auch können Sie dieselbe Integration für mehrere Routen wiederverwenden.

Route mit der API Gateway-Konsole erstellen

So erstellen Sie eine Route über die API Gateway-Konsole:

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie bei Route key (Routenschlüssel) den Schlüsselnamen ein. Sie können die vordefinierten Routen (`$connect`, `$disconnect` und `$default`) oder eine benutzerdefinierte Route erstellen.

Note

Wenn Sie eine benutzerdefinierte Route erstellen, verwenden Sie nicht das `$`-Präfix im Routenschlüsselnamen. Dieses Präfix ist für vordefinierte Routen reserviert.

4. Wählen und konfigurieren Sie den Integrationstyp für die Route. Weitere Informationen finden Sie unter [the section called “Richten Sie mit der WebSocket API Gateway-Konsole eine API-Integrationsanfrage ein”](#).

Erstellen Sie eine Route mit dem AWS CLI

Der folgende [create-route](#)-Befehl erstellt eine Route:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccdde --route-key $default
```

Die Ausgabe sieht wie folgt aus:

```
{
  "ApiKeyRequired": false,
  "AuthorizationType": "NONE",
  "RouteKey": "$default",
  "RouteId": "1122334"
}
```

Angeben der Routenanforderungs-Einstellungen für **\$connect**

Wenn Sie die `$connect`-Route für Ihre API einrichten, sind die folgenden optionalen Einstellungen verfügbar, um die Autorisierung für Ihre API zu ermöglichen. Weitere Informationen finden Sie unter [the section called “Die Route `\$connect`”](#).

- **Authorization (Autorisierung):** Wenn keine Autorisierung erforderlich ist, können Sie `NONE` angeben. Geben Sie andernfalls Folgendes ein:
 - `AWS_IAM` um den Zugriff auf Ihre API mithilfe von AWS Standard-IAM-Richtlinien zu kontrollieren.
 - `CUSTOM`, um die Autorisierung für eine API durch Angabe einer zuvor von Ihnen erstellten Lambda-Genehmigerfunktion zu implementieren. Der Autorisierer kann sich in Ihrem eigenen AWS Konto oder einem anderen Konto befinden. AWS Weitere Informationen über Lambda-Genehmiger finden Sie unter [API Gateway-Lambda-Genehmiger verwenden](#).

Note

In der API Gateway-Konsole ist die Einstellung `CUSTOM` erst sichtbar, nachdem Sie eine Genehmigerfunktion wie unter [the section called “Lambda-Genehmiger konfigurieren \(Konsole\)”](#) beschrieben eingerichtet haben.

Important

Die Einstellung von **Authorization (Autorisierung)** wird auf die gesamte API angewendet, nicht nur auf die `$connect`-Route. Die `$connect`-Route schützt die übrigen Routen, da sie für jede Verbindung aufgerufen wird.

- API-Schlüssel erforderlich: Sie können für die \$connect-Route einer API optional einen API-Schlüssel verlangen. Sie können API-Schlüssel zusammen mit Nutzungsplänen verwenden, um den Zugriff auf Ihre zu kontrollieren und zu verfolgen. APIs Weitere Informationen finden Sie unter [the section called “Nutzungspläne”](#).

\$connect-Routenanforderung über die API Gateway-Konsole einrichten

So richten Sie die \$connect Routenanforderung für eine WebSocket API mithilfe der API-Gateway-Konsole ein:

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie unter Routes (Routen) \$connect aus oder erstellen Sie eine \$connect-Route, indem Sie wie folgt vorgehen [the section called “Route mit der API Gateway-Konsole erstellen”](#).
3. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Authorization (Autorisierung) einen Autorisierungstyp aus.
5. Um eine API für die \$connect-Route anzufordern, wählen Sie Require API key (API-Schlüssel erforderlich) aus.
6. Wählen Sie Änderungen speichern aus.

Richten Sie Routenantworten für WebSocket APIs in API Gateway ein

WebSocket Routen können für bidirektionale oder unidirektionale Kommunikation konfiguriert werden. API-Gateway übergibt die Backend-Antwort nicht über die Routenantwort, es sei denn, Sie haben eine Routenantwort eingerichtet.

Note

Sie können nur die \$default Routenantwort für definieren. WebSocket APIs Sie können eine Integrationsantwort verwenden, um die Antwort von einem Backend-Service zu manipulieren. Weitere Informationen finden Sie unter [the section called “Übersicht über Integrationsantworten”](#).

Sie können Routenantworten und Antwortauswahlausdrücke mithilfe der API Gateway Gateway-Konsole oder des AWS CLI oder eines AWS SDK konfigurieren.

Weitere Informationen über Routenantwort-Auswahlausdrücke finden Sie unter [the section called “”](#).

Themen

- [Routenantwort mit der API Gateway-Konsole einrichten](#)
- [Richten Sie eine Routenantwort ein, indem Sie AWS CLI](#)

Routenantwort mit der API Gateway-Konsole einrichten

Nachdem Sie eine WebSocket API erstellt und eine Proxy-Lambda-Funktion an die Standardroute angehängt haben, können Sie die Routenantwort mit der API Gateway Gateway-Konsole einrichten:

1. Melden Sie sich bei der API Gateway Gateway-Konsole an und wählen Sie eine WebSocket API mit einer Proxy-Lambda-Funktionsintegration auf der `$default` Route aus.
2. Wählen Sie unter Routes (Routen) die `$default`-Route aus.
3. Wählen Sie Enable two-way communication (Bidirektionale Kommunikation aktivieren).
4. Klicken Sie auf Deploy API.
5. Stellen Sie Ihre API für eine Stufe bereit.

Stellen Sie mit dem folgenden [wscat](#)-Befehl eine Verbindung mit Ihrer API her. Mehr über `wscat` erfahren Sie unter [the section called “Wird verwendetwscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Drücken Sie die Eingabetaste, um die Standardroute aufzurufen. Der Hauptteil Ihrer Lambda-Funktion sollte zurückgegeben werden.

Richten Sie eine Routenantwort ein, indem Sie AWS CLI

Der folgende [create-route-response](#)Befehl erstellt eine Routenantwort für die `$default` Route. Sie können die API-ID und Routing-ID identifizieren, indem Sie die Befehle [get-apis](#) und [get-routes](#) verwenden.

```
aws apigatewayv2 create-route-response \
```

```
--api-id aabbccdde \  
--route-id 1122334 \  
--route-response-key '$default'
```

Die Ausgabe sieht wie folgt aus:

```
{  
  "RouteResponseId": "abcdef",  
  "RouteResponseKey": "$default"  
}
```

Richten Sie eine **\$connect** Route ein, für die ein WebSocket Unterprotokoll erforderlich ist

Clients können das `Sec-WebSocket-Protocol` Feld verwenden, um während der Verbindung zu Ihrer WebSocket API ein [WebSocket Unterprotokoll](#) anzufordern. Sie können eine Integration für die \$connect-Route einrichten, um Verbindungen nur zuzulassen, wenn ein Client ein von Ihrer API unterstütztes Unterprotokoll anfordert.

Die folgende Lambda-Beispielfunktion gibt den `Sec-WebSocket-Protocol`-Header an Clients zurück. Die Funktion stellt nur dann eine Verbindung zu Ihrer API her, wenn der Client das `myprotocol`-Unterprotokoll angibt.

Eine CloudFormation Vorlage, die diese Beispiel-API- und Lambda-Proxyintegration erstellt, finden Sie unter [ws-subprotocol.yaml](#).

```
export const handler = async (event) => {  
  if (event.headers !== undefined) {  
    const headers = toLowerCaseProperties(event.headers);  
  
    if (headers['sec-websocket-protocol'] !== undefined) {  
      const subprotocolHeader = headers['sec-websocket-protocol'];  
      const subprotocols = subprotocolHeader.split(',');  
  
      if (subprotocols.indexOf('myprotocol') >= 0) {  
        const response = {  
          statusCode: 200,  
          headers: {  
            "Sec-WebSocket-Protocol" : "myprotocol"  
          }  
        }  
      }  
    }  
  }  
}
```

```
        };
        return response;
    }
}

const response = {
    statusCode: 400
};

return response;
};

function toLowerCaseProperties(obj) {
    var wrapper = {};
    for (var key in obj) {
        wrapper[key.toLowerCase()] = obj[key];
    }
    return wrapper;
}
```

Sie können [wscat](#) verwenden, um zu testen, ob Ihre API Verbindungen nur zulässt, wenn ein Client ein von Ihrer API unterstütztes Unterprotokoll anfordert. Die folgenden Befehle verwenden das `-s`-Flag, um während der Verbindung Unterprotokolle anzugeben.

Mit dem folgenden Befehl wird versucht, eine Verbindung mit einem nicht unterstützten Unterprotokoll herzustellen. Da der Client das `chat1`-Unterprotokoll angegeben hat, gibt die Lambda-Integration einen 400-Fehler zurück und die Verbindung ist nicht erfolgreich.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

Der folgende Befehl enthält ein unterstütztes Unterprotokoll in der Verbindungsanforderung. Die Lambda-Integration ermöglicht die Verbindung.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Weitere Informationen zum Aufrufen WebSocket APIs finden Sie unter. [Aufrufen WebSocket APIs](#)

Steuern und verwalten Sie den Zugriff WebSocket APIs auf das API Gateway

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre WebSocket API.

Sie können die folgenden Mechanismen für die Authentifizierung und Autorisierung verwenden:

- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen. Mithilfe von IAM-Rollen und -Richtlinien können Sie steuern, wer Ihre APIs Rollen erstellen und verwalten darf und wer sie aufrufen kann. Weitere Informationen finden Sie unter [Steuern Sie den Zugriff auf WebSocket APIs mit IAM-Autorisierung](#).
- IAM-Tags können zusammen mit IAM-Richtlinien verwendet werden, um den Zugriff zu steuern. Weitere Informationen finden Sie unter [Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden](#).
- Lambda-Autorisierer sind Lambda-Funktionen, die den Zugriff auf kontrollieren. APIs Weitere Informationen finden Sie unter [Steuern Sie den Zugriff WebSocket APIs mit AWS Lambda REQUEST-Autorisierern](#).

Um Ihre Sicherheitslage zu verbessern, empfehlen wir Ihnen, auf allen Ihren Routen einen Autorisierer für die `$connect` Route zu konfigurieren. WebSocket APIs Dies kann erforderlich sein, um verschiedene Compliance-Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

Themen

- [Steuern Sie den Zugriff auf WebSocket APIs mit IAM-Autorisierung](#)
- [Steuern Sie den Zugriff WebSocket APIs mit AWS Lambda REQUEST-Autorisierern](#)

Steuern Sie den Zugriff auf WebSocket APIs mit IAM-Autorisierung

Die IAM-Autorisierung in WebSocket APIs ähnelt der für [REST APIs](#), mit den folgenden Ausnahmen:

- Die Aktion `execute-api` unterstützt `ManageConnections` zusätzlich zu vorhandenen Aktionen (`Invoke`, `InvalidateCache`). `ManageConnections` steuert den Zugriff auf die API `@connections`.
- WebSocket Routen verwenden ein anderes ARN-Format:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- Die @connections API verwendet dasselbe ARN-Format wie REST APIs:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

Important

Wenn Sie die [IAM-Autorisierung](#) verwenden, müssen Sie Anforderungen mit [Signature Version 4 \(SigV4\)](#) signieren.

Sie könnten z. B. die folgende Richtlinie für den Client einrichten. Dieses Beispiel ermöglicht allen Benutzern das Senden einer Nachricht (Invoke) für alle Routen, außer für eine geheime Route in der Stufe prod, und hindert alle Benutzer am Senden einer Nachricht zurück an verbundene Clients (ManageConnections) für alle Stufen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/secret"
      ]
    }
  ]
}
```

```

    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:ManageConnections"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111122223333:api-id/*"
      ]
    }
  ]
}

```

Steuern Sie den Zugriff WebSocket APIs mit AWS Lambda REQUEST-Autorisierern

Eine Lambda-Autorisierungsfunktion in WebSocket APIs ähnelt der für [REST APIs](#), mit den folgenden Ausnahmen:

- Sie können für die `$connect`-Route nur eine Lambda-Genehmiger-Funktion verwenden.
- Sie können keine Pfadvariablen (`event.pathParameters`) verwenden, da der Pfad fest ist.
- `event.methodArn` unterscheidet sich von seinem REST API-Äquivalent, da es über keine HTTP-Methode verfügt. Im Fall von `$connect` endet `methodArn` mit `"$connect"`:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
```

- Die Kontextvariablen in `event.requestContext` unterscheiden sich von denen für REST APIs

Das folgende Beispiel zeigt eine Eingabe für einen REQUEST Autorisierer für eine WebSocket API:

```

{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/default/$connect",
  "headers": {
    "Connection": "upgrade",
    "content-length": "0",
    "HeaderAuth1": "headerValue1",
    "Host": "abcdef123.execute-api.us-east-1.amazonaws.com",
    "Sec-WebSocket-Extensions": "permessage-deflate; client_max_window_bits",
    "Sec-WebSocket-Key": "...",

```

```
    "Sec-WebSocket-Version": "13",
    "Upgrade": "websocket",
    "X-Amzn-Trace-Id": "...",
    "X-Forwarded-For": "...",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "multiValueHeaders": {
    "Connection": [
      "upgrade"
    ],
    "content-length": [
      ""
    ],
    "HeaderAuth1": [
      "headerValue1"
    ],
    "Host": [
      "abcdef123.execute-api.us-east-1.amazonaws.com"
    ],
    "Sec-WebSocket-Extensions": [
      "permessage-deflate; client_max_window_bits"
    ],
    "Sec-WebSocket-Key": [
      "..."
    ],
    "Sec-WebSocket-Version": [
      "13"
    ],
    "Upgrade": [
      "websocket"
    ],
    "X-Amzn-Trace-Id": [
      "..."
    ],
    "X-Forwarded-For": [
      "..."
    ],
    "X-Forwarded-Port": [
      "443"
    ],
    "X-Forwarded-Proto": [
      "https"
    ]
  ]
}
```

```

},
"queryStringParameters": {
  "QueryString1": "queryValue1"
},
"multiValueQueryStringParameters": {
  "QueryString1": [
    "queryValue1"
  ]
},
"stageVariables": {},
"requestContext": {
  "routeKey": "$connect",
  "eventType": "CONNECT",
  "extendedRequestId": "...",
  "requestTime": "19/Jan/2023:21:13:26 +0000",
  "messageDirection": "IN",
  "stage": "default",
  "connectedAt": 1674162806344,
  "requestTimeEpoch": 1674162806345,
  "identity": {
    "sourceIp": "..."
  },
  "requestId": "...",
  "domainName": "abcdef123.execute-api.us-east-1.amazonaws.com",
  "connectionId": "...",
  "apiId": "abcdef123"
}
}

```

Die folgende Lambda-Autorisierungsfunktion ist eine WebSocket Version der Lambda-Autorisierungsfunktion für REST in: APIs [the section called “Weitere Beispiele für Lambda-Genehmigerfunktionen”](#)

Node.js

```

// A simple REQUEST authorizer example to demonstrate how to use request
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header and QueryString1 query
// parameter
// in the request context match the specified values of
// of 'headerValue1' and 'queryValue1' respectively.
export const handler = function(event, context, callback) {
  console.log('Received event:', JSON.stringify(event, null, 2));

```

```
// Retrieve request parameters from the Lambda function input:
var headers = event.headers;
var queryStringParameters = event.queryStringParameters;
var stageVariables = event.stageVariables;
var requestContext = event.requestContext;

// Parse the input for the parameter values
var tmp = event.methodArn.split(':');
var apiGatewayArnTmp = tmp[5].split('/');
var awsAccountId = tmp[4];
var region = tmp[3];
var ApiId = apiGatewayArnTmp[0];
var stage = apiGatewayArnTmp[1];
var route = apiGatewayArnTmp[2];

// Perform authorization to return the Allow policy for correct parameters and
// the 'Unauthorized' error, otherwise.
var authResponse = {};
var condition = {};
    condition.IpAddress = {};

if (headers.HeaderAuth1 === "headerValue1"
    && queryStringParameters.QueryString1 === "queryValue1") {
    callback(null, generateAllow('me', event.methodArn));
} else {
    callback(null, generateDeny('me', event.methodArn));
}
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'           '; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
    }
}
```

```

        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}

```

Python

```

# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
# parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json

def lambda_handler(event, context):
    print(event)

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')

```

```

awsAccountId = tmp[4]
region = tmp[3]
ApiId = apiGatewayArnTmp[0]
stage = apiGatewayArnTmp[1]
route = apiGatewayArnTmp[2]

# Perform authorization to return the Allow policy for correct parameters
# and the 'Unauthorized' error, otherwise.

authResponse = {}
condition = {}
condition['IpAddress'] = {}

if (headers['HeaderAuth1'] ==
    "headerValue1" and queryStringParameters["QueryString1"] ==
"queryValue1"):
    response = generateAllow('me', event['methodArn'])
    print('authorized')
    return json.loads(response)
else:
    response = generateDeny('me', event['methodArn'])
    print('unauthorized')
    return json.loads(response)

# Help function to generate IAM policy

def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,

```



```
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON

def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)

def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

[Gehen Sie genauso vor wie bei REST, um die vorherige Lambda-Funktion als REQUEST Autorisierungsfunktion für eine WebSocket API zu konfigurieren. APIs](#)

Um die \$connect-Route für die Verwendung dieses Lambda-Genehmigers in der Konsole zu konfigurieren, wählen Sie die \$connect-Route oder erstellen diese. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus. Wählen Sie im Dropdown-Menü Authorization (Autorisierung) Ihren Autorisierer aus und wählen Sie dann Save changes (Änderungen speichern) aus.

Zum Testen des Genehmigers müssen Sie eine neue Verbindung erstellen. Wenn der Genehmiger in \$connect geändert wird, hat dies keine Auswirkungen auf den bereits verbundenen Client. Wenn Sie eine Verbindung zu Ihrer WebSocket API herstellen, müssen Sie Werte für alle konfigurierten Identitätsquellen angeben. Beispielsweise können Sie eine Verbindung durch Senden einer gültigen Abfragezeichenfolge und eines gültigen Headers unter Verwendung von `wscat` wie im folgenden Beispiel herstellen:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString=queryValue1' -H HeaderAuth1:headerValue1
```

Bei dem Versuch, ohne einen gültigen Identitätswert eine Verbindung herzustellen, erhalten Sie eine 401-Antwort:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Integrationen für das WebSocket APIs API Gateway

Nach dem Einrichten einer API-Route müssen Sie diese in einen Endpunkt im Backend integrieren. Ein Backend-Endpunkt wird auch als Integrationsendpunkt bezeichnet und kann eine Lambda-Funktion, ein HTTP-Endpunkt oder eine AWS Serviceaktion sein. Die API-Integration beinhaltet eine Integrationsanforderung und eine Integrationsantwort.

In diesem Abschnitt erfahren Sie, wie Sie Integrationsanfragen und Integrationsantworten für Ihre WebSocket API einrichten.

Themen

- [Richten Sie eine WebSocket API-Integrationsanfrage in API Gateway ein](#)
- [Richten Sie eine WebSocket API-Integrationsantwort in API Gateway ein](#)

Richten Sie eine WebSocket API-Integrationsanfrage in API Gateway ein

Das Einrichten einer Integrationsanforderung umfasst die folgenden Schritte:

- Auswählen eines Routenschlüssels zum Integrieren im Backend.
- Angabe des aufzurufenden Backend-Endpunkts. WebSocket APIs unterstützt die folgenden Integrationstypen:
 - AWS_PROXY
 - AWS
 - HTTP_PROXY
 - HTTP
 - MOCK

Weitere Informationen zu Integrationstypen finden Sie [IntegrationType](#) in der API Gateway V2 REST API.

- Konfigurieren, wie die Routenanforderungsdaten bei Bedarf in Integrationsanforderungsdaten konvertiert werden, durch Angabe einer oder mehrerer Anforderungsvorlagen.

Richten Sie mit der WebSocket API Gateway-Konsole eine API-Integrationsanfrage ein

So fügen Sie mithilfe der API-Gateway-Konsole eine Integrationsanfrage zu einer Route in einer WebSocket API hinzu

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie unter Routes (Routen) die Route aus.
3. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Integration type (Integrationstyp) eine der folgenden Optionen:
 - Wählen Sie die Lambda-Funktion nur, wenn Ihre API in eine AWS Lambda Funktion integriert wird, die Sie bereits in diesem Konto oder in einem anderen Konto erstellt haben.

Um eine neue Lambda-Funktion in zu erstellen AWS Lambda, eine Ressourcenberechtigung für die Lambda-Funktion festzulegen oder andere Lambda-Serviceaktionen auszuführen, wählen Sie AWS stattdessen Service.

- Wählen Sie HTTP, wenn Ihre API in einen vorhandenen HTTP-Endpunkt integriert wird. Weitere Informationen finden Sie unter [HTTP-Integrationen für REST APIs in API Gateway](#).
 - Wählen Sie Mock, wenn Sie API-Antworten direkt von API Gateway generieren möchten, ohne dass ein Integrations-Backend erforderlich ist. Weitere Informationen finden Sie unter [Scheinintegrationen für REST APIs in API Gateway](#).
 - Wählen Sie AWS Service, wenn Ihre API in einen Dienst integriert werden soll. AWS
 - Wählen Sie VPC Link (VPC-Link), wenn Ihre API einen VpcLink als privaten Integrations-Endpunkt verwenden wird. Weitere Informationen finden Sie unter [Richten Sie eine private Integration ein](#).
5. Wenn Sie Lambda Function (Lambda-Funktion) wählen, gehen Sie wie folgt vor:
 - a. Markieren Sie unter Use Lambda Proxy integration (Lambda-Proxy-Integration verwenden) das Kontrollkästchen, wenn Sie die [Lambda-Proxy-Integration](#) oder die [kontoübergreifende Lambda-Proxy-Integration](#) verwenden möchten.
 - b. Geben Sie für Lambda Function (Lambda-Funktion) die Funktion auf eine der folgenden Arten an:

- Wenn sich Ihre Lambda-Funktion in demselben Konto befindet, geben Sie den Funktionsnamen ein und wählen Sie dann die Funktion aus der Dropdown-Liste aus.

 Note

Der Funktionsname kann optional seinen Alias oder seine Versionsangabe, wie in HelloWorld, HelloWorld:1 oder HelloWorld:alpha, enthalten.

- Wenn sich die Funktion in einem anderen Konto befindet, geben Sie den ARN für die Funktion ein.
- c. Wenn Sie den Timeout-Standardwert von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.
6. Befolgen Sie bei Wahl von HTTP die Anweisungen in Schritt 4 von [the section called “Einrichten einer API-Integrationsanforderung mit der Konsole”](#).
 7. Befolgen Sie bei Wahl von Mock (Pseudo) die Anweisungen unter dem Schritt Request Templates (Vorlagen anfordern).
 8. Befolgen Sie bei Wahl von AWS -Service die Anweisungen in Schritt 6 von [the section called “Einrichten einer API-Integrationsanforderung mit der Konsole”](#).
 9. Wenn Sie VPC Link (VPC-Link) ausgewählt haben, gehen Sie wie folgt vor:
 - a. Markieren Sie für VPC proxy integration (Proxy-Integration verwenden) das Kontrollkästchen, wenn Anforderungen über einen Proxy an Ihren VPCLink-Endpunkt gesendet werden sollen.
 - b. Wählen Sie als HTTP method die HTTP-Methode aus, die am ehesten der Methode im HTTP-Backend entspricht.
 - c. Wählen Sie in der Dropdownliste VPC link (VPC-Link) einen VPC-Link aus. Sie können [Use Stage Variables] auswählen und `${stageVariables.vpcLinkId}` in das Textfeld unter der Liste eingeben.

Sie können die vpcLinkId-Stufenvariable nach der Bereitstellung der API für eine Stufe definieren und ihren Wert auf die ID des VpcLink festlegen.

- d. Geben Sie als Endpunkt-URL die URL des HTTP-Backends ein, das von dieser Integration verwendet werden soll.
 - e. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen 50 und 29000 Millisekunden ein.
10. Wählen Sie **Änderungen speichern** aus.
11. Führen Sie unter **Request Templates** (Vorlagen anfordern) die folgenden Schritte durch:
- a. Um einen **Template selection expression** (Vorlagen-Auswahlausdruck) einzugeben, wählen Sie unter **Request templates** (Vorlagen anfordern) die Option **Edit** (Bearbeiten).
 - b. Geben Sie einen **Template selection expression** (Vorlagen-Auswahlausdruck) ein. Verwenden Sie einen Ausdruck, nach dem API Gateway in der Nachrichtennutzlast sucht. Wenn er gefunden wird, wird er ausgewertet. Das Ergebnis ist ein Vorlagen-Schlüsselwert zur Auswahl der Datenzuweisungsvorlage, die auf die Daten in der Nachrichtennutzlast anzuwenden ist. Die Datenzuweisungsvorlage erstellen Sie im nächsten Schritt. Wählen Sie **Edit** (Bearbeiten) aus, um Ihre Änderungen zu speichern.
 - c. Wählen Sie **Create template** (Vorlage erstellen), um die Datenzuweisungsvorlage zu erstellen. Geben Sie für **Template key** (Vorlagenschlüssel) einen Vorlagenschlüsselwert zur Auswahl der Datenzuweisungsvorlage ein, die auf die Daten in der Nachrichtennutzlast anzuwenden ist. Definieren Sie anschließend eine Zuweisungsvorlage. Wählen Sie **Create template** (Vorlage erstellen) aus.

Weitere Informationen zu Vorlagen-Auswahlausdrücken finden Sie unter [the section called "Vorlagen-Auswahlausdrücke"](#).

Richten Sie eine Integrationsanfrage ein, indem Sie AWS CLI

Sie können eine Integrationsanfrage für eine Route in einer WebSocket API einrichten, indem Sie AWS CLI wie im folgenden Beispiel vorgehen, wodurch eine Scheinintegration erstellt wird:

1. Erstellen Sie eine Datei mit dem Namen `integration-params.json` und dem folgenden Inhalt:

```
{
  "PassthroughBehavior": "WHEN_NO_MATCH",
  "TimeoutInMillis": 29000,
  "ConnectionType": "INTERNET",
  "RequestTemplates": {
    "application/json": "{\"statusCode\":200}"
  },
  "IntegrationType": "MOCK"
}
```

2. Verwenden Sie den folgenden [create-integration](#)-Befehl, um die Mock-Integration zu erstellen.

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccdde --cli-input-json file://integration-params.json
```

Die Ausgabe sieht wie folgt aus:

```
{
  "PassthroughBehavior": "WHEN_NO_MATCH",
  "TimeoutInMillis": 29000,
  "ConnectionType": "INTERNET",
  "IntegrationResponseSelectionExpression": "${response.statuscode}",
  "RequestTemplates": {
    "application/json": "{\"statusCode\":200}"
  },
  "IntegrationId": "0abcdef",
  "IntegrationType": "MOCK"
}
```

Alternativ können Sie eine Integrationsanfrage für eine Proxyintegration einrichten, indem Sie den folgenden AWS CLI-Befehl verwenden.

1. Erstellen Sie eine Lambda-Funktion in der Lambda-Konsole und geben Sie ihr eine grundlegende Lambda-Ausführungsrolle.
2. Verwenden Sie den folgenden [create-integration](#)-Befehl, um eine Integration zu erstellen.

```
aws apigatewayv2 create-integration --api-id aabbccdde --integration-type
AWS_PROXY --integration-method POST --integration-uri arn:aws:apigateway:us-
east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
east-1:123412341234:function:simpleproxy-echo-e2e/invocations
```

Die Ausgabe sieht wie folgt aus:

```
{
```

```

    "PassthroughBehavior": "WHEN_NO_MATCH",
    "IntegrationMethod": "POST",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationUri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:simpleproxy-echo-e2e/invocations",
    "IntegrationId": "abcdefg",
    "IntegrationType": "AWS_PROXY"
}

```

Eingabeformat einer Lambda-Funktion für die Proxyintegration für WebSocket APIs

Bei der Lambda-Proxy-Integration ordnet API Gateway die gesamte Client-Anforderung dem event-Eingabeparameter der Backend-Lambda-Funktion zu. Im folgenden Beispiel sehen Sie die Struktur des Eingabeereignisses der \$connect-Route und des Eingabeereignisses der \$disconnect-Route, die das API Gateway an eine Lambda-Proxy-Integration sendet.

Input from the \$connect route

```

{
  headers: {
    Host: 'abcd123.execute-api.us-east-1.amazonaws.com',
    'Sec-WebSocket-Extensions': 'permessage-deflate; client_max_window_bits',
    'Sec-WebSocket-Key': '...',
    'Sec-WebSocket-Version': '13',
    'X-Amzn-Trace-Id': '...',
    'X-Forwarded-For': '192.0.2.1',
    'X-Forwarded-Port': '443',
    'X-Forwarded-Proto': 'https'
  },
  multiValueHeaders: {
    Host: [ 'abcd123.execute-api.us-east-1.amazonaws.com' ],
    'Sec-WebSocket-Extensions': [ 'permessage-deflate; client_max_window_bits' ],
    'Sec-WebSocket-Key': [ '...' ],
    'Sec-WebSocket-Version': [ '13' ],
    'X-Amzn-Trace-Id': [ '...' ],
    'X-Forwarded-For': [ '192.0.2.1' ],
    'X-Forwarded-Port': [ '443' ],
    'X-Forwarded-Proto': [ 'https' ]
  },
  requestContext: {
    routeKey: '$connect',
    eventType: 'CONNECT',

```

```

    extendedRequestId: 'ABCD1234=',
    requestTime: '09/Feb/2024:18:11:43 +0000',
    messageDirection: 'IN',
    stage: 'prod',
    connectedAt: 1707502303419,
    requestTimeEpoch: 1707502303420,
    identity: { sourceIp: '192.0.2.1' },
    requestId: 'ABCD1234=',
    domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
    connectionId: 'AAAA1234=',
    apiId: 'abcd1234'
  },
  isBase64Encoded: false
}

```

Input from the \$disconnect route

```

{
  headers: {
    Host: 'abcd1234.execute-api.us-east-1.amazonaws.com',
    'x-api-key': '',
    'X-Forwarded-For': '',
    'x-restapi': ''
  },
  multiValueHeaders: {
    Host: [ 'abcd1234.execute-api.us-east-1.amazonaws.com' ],
    'x-api-key': [ '' ],
    'X-Forwarded-For': [ '' ],
    'x-restapi': [ '' ]
  },
  requestContext: {
    routeKey: '$disconnect',
    disconnectStatusCode: 1005,
    eventType: 'DISCONNECT',
    extendedRequestId: 'ABCD1234=',
    requestTime: '09/Feb/2024:18:23:28 +0000',
    messageDirection: 'IN',
    disconnectReason: 'Client-side close frame status not set',
    stage: 'prod',
    connectedAt: 1707503007396,
    requestTimeEpoch: 1707503008941,
    identity: { sourceIp: '192.0.2.1' },

```



```
requestId: 'ABCD1234=',  
domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',  
connectionId: 'AAAA1234=',  
apiId: 'abcd1234'  
},  
isBase64Encoded: false  
}
```

Richten Sie eine WebSocket API-Integrationsantwort in API Gateway ein

Der folgende Abschnitt bietet einen kurzen Überblick über Integrationsantworten für WebSocket API und wie eine Integrationsantwort für eine WebSocket API eingerichtet wird.

Themen

- [Übersicht über Integrationsantworten](#)
- [Integrationsantworten für bidirektionale Kommunikation](#)
- [Integrationsantwort mit der API Gateway-Konsole einrichten](#)
- [Richten Sie eine Integrationsantwort mit dem ein AWS CLI](#)

Übersicht über Integrationsantworten

Die Integrationsantwort von API Gateway ist eine Möglichkeit, die Antwort von einem Backend-Service zu modellieren und zu manipulieren. Es gibt einige Unterschiede bei der Einrichtung einer REST-API und einer WebSocket API-Integrationsantwort, aber konzeptionell ist das Verhalten dasselbe.

WebSocket Routen können für bidirektionale oder unidirektionale Kommunikation konfiguriert werden.

- Wenn eine Route für bidirektionale Kommunikation konfiguriert ist, können Sie mit einer Integrationsantwort Transformationen für die zurückgegebene Nachrichtennutzlast konfigurieren, ähnlich wie bei Integrationsantworten für REST APIs
- Wenn eine Route für unidirektionale Kommunikation konfiguriert ist, wird unabhängig von der Konfiguration der Integrationsantwort nach der Verarbeitung der Nachricht keine Antwort über den WebSocket Kanal zurückgegeben.

API Gateway übergibt die Backend-Antwort nicht über die Routenantwort, es sei denn, Sie richten eine Routenantwort ein. Weitere Informationen zum Einrichten einer Routenreaktion finden Sie unter [the section called “Richten Sie WebSocket API-Routenantworten ein”](#).

Integrationsantworten für bidirektionale Kommunikation

Integrationen lassen sich in Proxy-Integrationen und Nicht-Proxy-Integrationen unterteilen.

 Important

Bei Proxy-Integrationen gibt API Gateway automatisch die Backend-Ausgabe als vollständigen Payload an den Aufrufer weiter. Es gibt keine Integrationsantwort.

Bei Nicht-Proxy-Integrationen müssen Sie mindestens eine Integrationsantwort einrichten:

- Im Idealfall sollte eine Ihrer Integrationsantworten als Catch-all-Methode dienen, wenn keine explizite Wahl vorgenommen werden kann. Dieser Standardfall wird durch Festlegen des Integrationsantwort-Schlüssels dargestellt `$default`.
- In allen anderen Fällen fungiert der Integrationsantwort-Schlüssel als regulärer Ausdruck. Er sollte dem Format folge `"/expression/"`.

Bei Nicht-Proxy-HTTP-Integrationen:

- API Gateway versucht, den HTTP-Statuscode der Backend-Antwort zuzuordnen. Der Integrationsantwort-Schlüssel fungiert in diesem Fall als regulärer Ausdruck. Wenn keine Übereinstimmung gefunden wird, dann wird `$default` als Integrationsantwort gewählt.
- Der Vorlagen-Auswahlausdruck funktioniert wie oben beschrieben identisch. Zum Beispiel:
 - `/2\d\d/`: Erfolgreiche Antworten empfangen und transformieren
 - `/4\d\d/`: Fehler (ungültige Anforderung) empfangen und transformieren
 - `$default`: Alle unerwarteten Antworten empfangen und transformieren

Weitere Informationen zu Vorlagen-Auswahlausdrücken finden Sie unter [the section called “Vorlagen-Auswahlausdrücke”](#).

Integrationsantwort mit der API Gateway-Konsole einrichten

So richten Sie mithilfe der API-Gateway-Konsole eine Antwort zur Routenintegration für eine WebSocket API ein:

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie Ihre WebSocket API und wählen Sie Ihre Route.
3. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Create integration response (Integrationsantwort erstellen) aus.
4. Geben Sie für Antwortschlüssel einen Wert ein, der nach Auswertung des Ausdrucks der Antwortauswahl im Antwortschlüssel der ausgehenden Nachricht zu finden ist. Sie können beispielsweise `/4\d\d/` eingeben, um Anforderungsfehler zu empfangen und umzuwandeln, oder Sie können `$default` eingeben, um alle Antworten zu empfangen und umzuwandeln, die dem Ausdruck für die Vorlagenauswahl entsprechen.
5. Geben Sie unter Template selection expression (Ausdruck für die Vorlagenauswahl) einen Auswahl Ausdruck ein, um die ausgehende Nachricht auszuwerten.
6. Wählen Sie Create response (Antwort erstellen) aus.
7. Sie können auch eine Zuordnungsvorlage definieren, um Transformationen Ihrer Payload für zurückgegebene Nachrichten zu konfigurieren. Wählen Sie Create template (Vorlage erstellen) aus.
8. Geben Sie einen Schlüsselnamen ein. Wenn Sie den Standardausdruck für die Vorlagenauswahl auswählen, geben Sie `\$default` ein.
9. Geben Sie unter Response template (Antwortvorlage) Ihre Zuordnungsvorlage in den Code-Editor ein.
10. Wählen Sie Create template (Vorlage erstellen) aus.
11. Wählen Sie Deploy API (API bereitstellen) aus, um Ihre API bereitzustellen.

Stellen Sie mit dem folgenden [wscat](#)-Befehl eine Verbindung mit Ihrer API her. Mehr über `wscat` erfahren Sie unter [the section called “Wird verwendetwscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Wenn Sie Ihre Route aufrufen, sollte die Payload der zurückgegebenen Nachricht zurückgegeben werden.

Richten Sie eine Integrationsantwort mit dem ein AWS CLI

Der folgende [create-integration-response](#) Befehl erstellt eine `$default` Integrationsantwort:

```
aws apigatewayv2 create-integration-response \  
  --api-id vaz7da96z6 \  
  --integration-id a1b2c3 \  
  --integration-response-key '$default'
```

Validierung für WebSocket APIs im API Gateway anfordern

Sie können API Gateway so konfigurieren, dass die Validierung einer Routenanforderung durchgeführt wird, bevor Sie mit der Integrationsanforderung fortfahren. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage fehl, ohne Ihr Backend aufzurufen, sendet eine Gateway-Antwort „Bad Request Body“ an den Client und veröffentlicht die Validierungsergebnisse in CloudWatch Logs. Die Verwendung der Validierung auf diese Weise reduziert unnötige Aufrufe an Ihr API-Backend.

Modell-Auswahlausdrücke

Sie können einen Modell-Auswahlausdruck verwenden, um Anforderungen innerhalb derselben Route dynamisch zu validieren. Die Modellvalidierung erfolgt, wenn Sie einen Modell-Auswahlausdruck für Proxy- oder Nicht-Proxy-Integrationen bereitstellen. Möglicherweise müssen Sie das `$default`-Modell als Fallback definieren, wenn kein passendes Modell gefunden wird. Wenn kein übereinstimmendes Modell vorhanden ist und `$default` nicht definiert ist, schlägt die Validierung fehl. Der Auswahlausdruck ähnelt `Route.ModelSelectionExpression` und wertet den Schlüssel für `Route.RequestModels` aus.

Wenn Sie eine Route für eine WebSocket API definieren, können Sie optional einen Modellauswahlausdruck angeben. Dieser Ausdruck wird zur Auswahl des Modells ausgewertet, das bei Eingang einer Anforderung für die Textvalidierung verwendet werden soll. Der Ausdruck wird auf einen der Einträge in den einer Route ausgewertet [requestmodels](#).

Ein Modell wird als [JSON-Schema](#) ausgedrückt und beschreibt die Datenstruktur des Anforderungstextes. Die Beschaffenheit dieses Auswahlausdrucks ermöglicht Ihnen, dynamisch das Modell auszuwählen, über das zur Laufzeit für eine bestimmte Route validiert werden soll. Weitere

Informationen, wie Sie ein Modell erstellen können, finden Sie unter [the section called “Datenmodelle für REST APIs”](#).

Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole

Im folgenden Beispiel wird gezeigt, wie Sie die Anforderungsvalidierung auf einer Route einrichten.

Zuerst erstellen Sie ein Modell und dann eine Route. Als Nächstes konfigurieren Sie die Anforderungsvalidierung für die Route, die Sie gerade erstellt haben. Abschließend stellen Sie Ihre API bereit und testen sie. Um dieses Tutorial abzuschließen, benötigen Sie eine WebSocket API `$request.body.action` als Routenauswahlausdruck und einen Integrationsendpunkt für Ihre neue Route.

Sie benötigen außerdem `wscat`, um eine Verbindung zu Ihrer API herzustellen. Weitere Informationen finden Sie unter [the section called “Wird verwendet wscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).

So erstellen Sie ein Modell

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie eine API aus. WebSocket
3. Klicken Sie im Navigationsbereich auf Models (Modelle).
4. Wählen Sie Modell erstellen aus.
5. Geben Sie unter Name **emailModel** ein.
6. Geben Sie für Content type (Inhaltstyp) **application/json** ein.
7. Geben Sie für Modellschema Folgendes ein:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type" : "object",
  "required" : [ "address"],
  "properties" : {
    "address": {
      "type": "string"
    }
  }
}
```

Für dieses Modell muss die Anfrage eine E-Mail-Adresse enthalten.

8. Wählen Sie Speichern.

In diesem Schritt erstellen Sie eine Route für Ihre WebSocket API.

So erstellen Sie eine Route

1. Klicken Sie im Hauptnavigationsbereich auf Routen.
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie in Route key (Routenschlüssel) **sendMessage** ein.
4. Wählen Sie einen Integrationstyp und geben Sie einen Integrationsendpunkt an. Weitere Informationen finden Sie unter [the section called "Integrationen"](#).
5. Wählen Sie Create route (Route erstellen) aus.

In diesem Schritt richten Sie die Anforderungsvalidierung für die sendMessage-Route ein.

Einrichten der Anforderungsvalidierung

1. Klicken Sie auf der Registerkarte Routenanforderung unter Routenanforderungseinstellungen auf Bearbeiten.
2. Geben Sie unter Modellauswahlausdruck **`${request.body.messageType}`** ein.

API Gateway verwendet die messageType-Eigenschaft zur Validierung der eingehenden Anforderung.

3. Klicken Sie auf Anforderungsmodell hinzufügen.
4. Geben Sie für Modellschlüssel **email** ein.
5. Wählen Sie unter Modell die Option emailModel aus.

API Gateway validiert eingehende Nachrichten mithilfe der messageType-Eigenschaft, die anhand dieses Modells auf email gesetzt ist.

Note

Wenn API Gateway den Modellauswahlausdruck nicht mit einem Modellschlüssel abgleichen kann, wird das `$default`-Modell ausgewählt. Wenn kein `$default`-Modell

vorhanden ist, schlägt die Validierung fehl. Für die Produktion empfehlen wir APIs, dass Sie ein `$default` Modell erstellen.

6. Wählen Sie **Änderungen speichern** aus.

In diesem Schritt stellen Sie Ihre API bereit und testen sie.

Bereitstellen und testen Ihrer API

1. Klicken Sie auf **Deploy API**.
2. Wählen Sie die gewünschte Stufe aus der Dropdown-Liste aus oder geben Sie den Namen einer neuen Stufe ein.
3. Wählen Sie **Bereitstellen**.
4. Klicken Sie im Hauptnavigationsbereich auf **Stufen**.
5. Kopieren Sie die WebSocket URL Ihrer API. Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.
6. Öffnen Sie ein neues Terminal. Führen Sie den Befehl `wscat` mit den folgenden Parametern aus.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Testen Sie Ihre API mit dem folgenden Befehl.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId": "ABCD1=234",  
  "requestId": "EFGH="}
```

Die Anforderung schlägt in API Gateway fehl.

Verwenden Sie den folgenden Befehl, um eine gültige Anforderung an Ihre API zu senden.

```
{"action": "sendMessage", "messageType": "email", "address":  
  "mary_major@example.com"}
```

Datentransformationen für WebSocket APIs im API Gateway

In API Gateway kann die Methodenanforderung einer WebSocket API eine Nutzlast in einem anderen Format als die entsprechende Nutzlast der Integrationsanfrage annehmen, wie es im Backend erforderlich ist. Das Backend wiederum kann eine Integrationsantwortnutzlast zurückgeben, die nicht der Methodenantwortnutzlast entspricht, die das Frontend erwartet.

In Amazon API Gateway können Sie Zuweisungsvorlagen verwenden, um die Nutzdaten einer Methodenanforderung der entsprechenden Integrationsanforderung zuzuordnen bzw. die Nutzdaten einer Integrationsanforderung der entsprechenden Methodenantwort zuzuordnen. Sie erstellen eine Zuweisungsvorlage und geben einen Vorlagen-Auswahlausdruck an, um zu bestimmen, welche Vorlage für die erforderlichen Datentransformationen verwendet werden soll.

Sie können Datenzuordnungen verwenden, um Daten aus einer [Routenanforderung](#) einer Backend-Integration zuzuordnen. Weitere Informationen hierzu finden Sie unter [the section called “Datenzuordnung”](#).

Zuweisungsvorlagen und Modelle

[Eine Mapping-Vorlage ist ein in Velocity Template Language \(VTL\) ausgedrücktes Skript, das mithilfe von Ausdrücken auf die Nutzlast angewendet wird. JSONPath](#) Weitere Informationen über API Gateway-Zuordnungsvorlagen finden Sie unter [Transformationen mit Zuweisungsvorlagen für REST-APIs in API Gateway](#).

Die Nutzlast kann ein dem [JSON-Schema Entwurf 4](#) entsprechendes Datenmodell umfassen. Sie müssen kein Modell definieren, um eine Zuweisungsvorlage zu erstellen. Ein Modell kann Ihnen jedoch bei der Erstellung einer Vorlage helfen, da API Gateway einen Vorlagenentwurf auf der Grundlage eines bereitgestellten Modells generiert. Weitere Informationen über API Gateway-Modelle finden Sie unter [Datenmodelle für REST APIs](#).

Vorlagen-Auswahlausdrücke

[Um eine Payload mit einer Mapping-Vorlage zu transformieren, geben Sie in einer Integrationsanfrage oder Integrationsantwort einen Ausdruck für die WebSocket API-Vorlagenauswahl an.](#) Durch Auswerten dieses Ausdrucks wird die Eingabe- oder Ausgabevorlage (falls vorhanden) bestimmt, die zum Transformieren des Anforderungstexts in den Integrationsanforderungstext (über eine Eingabevorlage) oder des Integrationsantworttext in den Routenantworttext (über eine Ausgabevorlage) verwendet werden soll.

`Integration.TemplateSelectionExpression` unterstützt `${request.body.jsonPath}` und statische Werte.

`IntegrationResponse.TemplateSelectionExpression` unterstützt `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` und statische Werte.

Integrationsantwort-Auswahlausdrücke

Wenn Sie [eine Integrationsantwort für eine WebSocket API einrichten](#), können Sie optional einen Auswahlausdruck für eine Integrationsantwort angeben. Dieser Ausdruck bestimmt, welche [IntegrationResponse](#) ausgewählt werden sollte, wenn eine Integration zurückgegeben wird. Der Wert dieses Ausdrucks wird zurzeit durch API Gateway eingeschränkt, wie nachfolgend definiert. Beachten Sie, dass dieser Ausdruck nur für Nicht-Proxy-Integrationen relevant ist. Eine Proxy-Integration übergibt die Antwortnutzlast einfach ohne Modellierung oder Modifikation an die aufrufende Methode zurück.

Im Gegensatz zu den anderen vorhergehenden Auswahlausdrücken unterstützt dieser Ausdruck derzeit ein Musterabgleich-Format. Der Ausdruck sollte in Schrägstriche verpackt werden.

Derzeit ist der Wert je nach fixier [integrationType](#):

- Für Lambda-basierte Integrationen lautet er `$integration.response.body.errorMessage`.
- Für HTTP- und MOCK-Integrationen ist dies `$integration.response.statuscode`.
- Für HTTP_PROXY und AWS_PROXY wird der Ausdruck nicht verwendet, da Sie anfordern, dass die Nutzlast über die aufrufende Methode übergeben wird.

Datenzuordnung für WebSocket APIs in API Gateway einrichten

Mit der Datenzuordnung können Sie Daten aus einer [Routenanforderung](#) einer Backend-Integration zuordnen.

Note

Die Datenzuordnung für wird in der WebSocket APIs nicht unterstützt AWS-Managementkonsole. Sie müssen das AWS CLI, oder ein SDK verwenden AWS CloudFormation, um die Datenzuordnung zu konfigurieren.

Themen

- [Zuordnen von Routenanforderungsdaten zu Integrationsanforderungsparametern](#)
- [Beispiele](#)

Zuordnen von Routenanforderungsdaten zu Integrationsanforderungsparametern

Integrationsanforderungsparameter können aus beliebigen definierten Routenanforderungsparametern, dem Anforderungstext [context](#) oder [stage](#)-Variablen und statischen Werten zugeordnet werden.

In der folgenden Tabelle sehen Sie Datenzuordnungsausdrücke für Integrationsanfragen. In der Tabelle ist *PARAM_NAME* der Name eines Routenanforderungsparameters des angegebenen Parametertyps. Es muss mit dem regulären Ausdruck übereinstimmen `^[a-zA-Z0-9._$-]+$`. *JSONPath_EXPRESSION* ist ein JSONPath Ausdruck für ein JSON-Feld des Hauptteils der Anfrage.

Zugewiesene Datenquelle	Mapping-Ausdruck
Anforderungsabfragezeichenfolge (wird nur für die \$connect-Route unterstützt)	<code>route.request.querystring. <i>PARAM_NAME</i></code>
Anforderungs-Header (wird nur für die \$connect-Route unterstützt)	<code>route.request.header. <i>PARAM_NAME</i></code>
Abfragezeichenfolge mit mehreren Werten (nur für die \$connect-Route unterstützt)	<code>route.request.multivaluedquerystring. <i>PARAM_NAME</i></code>
Anforderungsheader mit mehreren Werten (nur für die \$connect-Route unterstützt)	<code>route.request.multivaluedheader. <i>PARAM_NAME</i></code>
Anforderungstext	<code>route.request.body. <i>JSONPath_EXPRESSION</i></code>
Stufenvariablen	<code>stageVariables. <i>VARIABLE_NAME</i></code>
Kontextvariablen	<code>context. <i>VARIABLE_NAME</i></code> , wobei die Variable zu den unterstützten Kontextvariablen gehören muss.

Zugewiesene Datenquelle	Mapping-Ausdruck
Statischer Wert	' <i>STATIC_VALUE</i> ' . Das <i>STATIC_VALUE</i> ist ein Zeichenkettenliteral und muss in einfache Anführungszeichen eingeschlossen werden.

Achten Sie beim Erstellen einer Datenzuordnung auf das AWS CLI richtige Format für die Verwendung von Literalen mit Zeichenfolgen in. AWS CLI Weitere Informationen finden Sie unter [Verwenden von Zeichenfolgen mit Anführungszeichen in der AWS CLI](#) im Benutzerhandbuch zur AWS Command Line Interface .

Beispiele

In den folgenden AWS CLI Beispielen werden Datenzuordnungen konfiguriert. Eine CloudFormation Beispielvorlage finden Sie unter. [websocket-data-mapping.yaml](#)

Zuordnen der ConnectionID eines Clients zu einem Header in einer Integrationsanforderung

Der folgende [update-integration](#)-Befehl weist die `connectionId` eines Clients dem `connectionId`-Header in der Anfrage an eine Backend-Integration zu:

```
aws apigatewayv2 update-integration \  
  --integration-id abc123 \  
  --api-id a1b2c3d4 \  
  --request-parameters  
  'integration.request.header.connectionId='context.connectionId'
```

Zuordnen eines Abfragezeichenfolgenparameters zu einem Header in einer Integrationsanforderung

Das folgende Beispiel weist einen `authToken`-Abfragezeichenfolgenparameter dem `authToken`-Header in der Integrationsanforderung zu.

1. Verwenden Sie den folgenden [update-route](#)-Befehl, um den `authToken`-Abfragezeichenfolgenparameter zu den Anforderungsparametern der Route hinzuzufügen.

```
aws apigatewayv2 update-route --route-id 0abcdef \  
  --api-id a1b2c3d4 \  
  --request-parameters '{"route.request.querystring.authToken": {"Required":  
false}}'
```

2. Verwenden Sie den folgenden [update-integration](#)-Befehl, um den Abfragezeichenfolgenparameter dem authToken-Header in der Anfrage an die Backend-Integration zuzuweisen.

```
aws apigatewayv2 update-integration \  
  --integration-id abc123 \  
  --api-id a1b2c3d4 \  
  --request-parameters  
  'integration.request.header.authToken='route.request.querystring.authToken'
```

3. (Optional) Gehen Sie bei Bedarf wie folgt vor, [delete-route-request-parameter](#) um den authToken Abfragezeichenfolgenparameter aus den Anforderungsparametern der Route zu löschen.

```
aws apigatewayv2 delete-route-request-parameter \  
  --route-id 0abcdef \  
  --api-id a1b2c3d4 \  
  --request-parameter-key 'route.request.querystring.authToken'
```

WebSocket Referenz zur API-Zuordnungsvorlage für API Gateway

In diesem Abschnitt werden die Variablen zusammengefasst, die derzeit WebSocket APIs in API Gateway unterstützt werden.

Parameter	Beschreibung
<code>\$context.connectionId</code>	Eine eindeutige ID für die Verbindung, die für einen Rückruf an den Client verwendet werden kann.
<code>\$context.connectedAt</code>	Die Epoch -formatierte Verbindungszeit.
<code>\$context.domainName</code>	Ein Domainname für die WebSocket API. Dies kann für einen Rückruf an den Client (anstelle eines hardcodierten Wertes) verwendet werden.

Parameter	Beschreibung
<code>\$context.eventType</code>	Der Ereignistyp: CONNECT, MESSAGE oder DISCONNECT .
<code>\$context.messageId</code>	Eine eindeutige serverseitige ID für eine Nachricht. Nur verfügbar, wenn der <code>\$context.eventType</code> MESSAGE ist.
<code>\$context.routeKey</code>	Der ausgewählte Routenschlüssel.
<code>\$context.requestId</code>	Entspricht <code>\$context.extendedRequestId</code> .
<code>\$context.extendedRequestId</code>	Eine automatisch generierte ID für den API-Aufruf, die weitere nützliche Informationen für das Debugging/die Fehlerbehebung enthält.
<code>\$context.apiId</code>	Die ID, die API Gateway Ihrer API zuweist.
<code>\$context.authorizer.principalId</code>	Die ID des Prinzipalbenutzer, die dem vom Client gesendeten und von einer Lambda-Funktion eines API Gateway-Lambda-Genehmigers (früher "benutzerdefinierter Genehmiger") zurückgegebenen Token zugeordnet ist.

Parameter	Beschreibung
<code>\$context.authorizer. <i>property</i></code>	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der context-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende context-Zuweisung zurück: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Dann gibt der Aufruf von <code>\$context.authorizer.key</code> die Zeichenfolge "value" zurück, der Aufruf von <code>\$context.authorizer.numKey</code> gibt die Zeichenfolge "1" zurück und der Aufruf von <code>\$context.authorizer.boolKey</code> gibt den Wert "true" zurück.
<code>\$context.error.messageString</code>	Der Wert von <code>\$context.error.message</code> in Anführungszeichen, d. h. " <code>\$context.error.message</code> ".
<code>\$context.error.validationErrorString</code>	Eine Zeichenfolge mit einer detaillierten Validierungs-Fehlermeldung.
<code>\$context.identity.accountId</code>	Die der Anfrage zugeordnete AWS Konto-ID.
<code>\$context.identity.apiKey</code>	Der API-Besitzerschlüssel, der der schlüsselfähigen API-Anforderung zugewiesen ist.
<code>\$context.identity.apiKeyId</code>	Die API-Schlüssel-ID, die der schlüsselfähigen API-Anforderung zugewiesen ist.

Parameter	Beschreibung
<code>\$context.identity.caller</code>	Die Prinzipal-ID des Aufrufers, von dem die Anforderung stammt.
<code>\$context.identity.cognitoAuthenticationProvider</code>	Eine durch Komma getrennte Liste aller Amazon-Cognito-Authentifizierungsanbieter, die vom anfordernden Aufrufer verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, <code>cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Weitere Informationen zu verfügbaren Amazon-Cognito-Authentifizierungsanbietern finden Sie unter Verbundidentitäten verwenden im Amazon-Cognito-Entwicklerhandbuch.
<code>\$context.identity.cognitoAuthenticationType</code>	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind <code>authenticated</code> für authentifizierte Identitäten und <code>unauthenticated</code> für nicht authentifizierte Identitäten.
<code>\$context.identity.cognitoIdentityId</code>	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Beschreibung
<code>\$context.identity.cognitoId</code> <code>entityPoolId</code>	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.sourceIp</code>	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
<code>\$context.identity.user</code>	Die Prinzipal-ID des Benutzers, von dem die Anforderung stammt.
<code>\$context.identity.userAgent</code>	Der Benutzeragent des API-Aufrufers.
<code>\$context.identity.userArn</code>	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung.
<code>\$context.requestTime</code>	Die Anforderungszeit im CLF -Format (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	Die Anforderungszeit im Epoch -Format in Millisekunden.
<code>\$context.stage</code>	Die Bereitstellungsstufe des API-Aufrufs (z. B. "Beta" oder "Prod").
<code>\$context.status</code>	Der Antwortstatus.
<code>\$input.body</code>	Gibt die Raw-Nutzlast als Zeichenfolge zurück.

Parameter	Beschreibung
<code>\$input.json(x)</code>	Diese Funktion wertet einen JSONPath Ausdruck aus und gibt die Ergebnisse als JSON-Zeichenfolge zurück. Beispielsweise gibt <code>\$input.json('\$.pets ')</code> eine JSON-Zeichenfolge zurück, die die "Pets"-Struktur abbildet. Weitere Informationen zu JSONPath, finden Sie unter JSONPath oder JSONPath für Java.

Parameter	Beschreibung
<code>\$input.path(x)</code>	Nimmt einen JSONPath Ausdruck string (x) und gibt eine JSON-Objektdarstellung des Ergebnisses zurück. Dies ermöglicht einen nativen Zugriff auf Elemente der Nutzlast in Apache Velocity Template Language (VTL) und deren Bearbeitung. Beispiel: Der Ausdruck <code>\$input.path('\$.pets')</code> könnte das folgende Objekt zurückgeben: <pre>[{ "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 }]</pre> <code>\$input.path('\$.pets').count()</code> gibt "3" zurück. Weitere Informationen zu JSONPath Java finden Sie unter JSONPath oder JSONPath für Java.
<code>\$stageVariables. <variable_name></code>	<code><variable_name></code> steht für einen Stufenvariablenamen.

Parameter	Beschreibung
<code>\$stageVariables[' <variable_name> ']</code>	<code><variable_name></code> steht für einen beliebigen Stufenvariablenamen.
<code>\${stageVariables[' <variable_name>']}</code>	<code><variable_name></code> steht für einen beliebigen Stufenvariablenamen.
<code>\$util.escapeJavaScript()</code>	Escapiert die Zeichen in einer Zeichenfolge mithilfe von JavaScript Zeichenfolgenregeln.  Note Mit dieser Funktion werden alle einfachen Anführungszeichen (') durch Escape-Zeichen (\ ') geschützt. Allerdings sind diese durch Escape-Zeichen geschützten einfachen Anführungszeichen in JSON nicht zulässig. Sofern die Ausgabe dieser Funktion in einer JSON-Eigenschaft verwendet werden soll, müssen alle einfachen Anführungszeichen, die durch Escape-Zeichen geschützt sind (\ '), wieder in reguläre einfache Anführungszeichen (') geändert werden. Das wird im folgenden Beispiel veranschaulicht: <pre>\$util.escapeJavaScript(ript(<i>data</i>).replaceAll("\\'", ,"'")</pre>

Parameter	Beschreibung
<code>\$util.parseJson()</code>	Erhält das "stringify"-JSON-Objekt und gibt eine Objektdarstellung des Ergebnisses zurück. Mit dem Ergebnis dieser Funktion können Sie Elemente der Nutzlast, die in Apache Velocity Template Language (VTL) sind, aufrufen und bearbeiten. Angenommen, Sie haben folgende Nutzlast: <pre>{ "errorMessage": "{ \"key1\": \"var1\", \"key2\": { \"arr\": [1,2,3] } }" }</pre> Und verwenden die folgende Mapping-Vorlage: <pre>#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$.errorMessage'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> Dann erhalten Sie die folgende Ausgabe: <pre>{ "errorMessageObjKey2ArrVal" : 1 }</pre>
<code>\$util.urlEncode()</code>	Konvertiert eine Zeichenfolge in das Format „application/x-www-form-urlencoded“.
<code>\$util.urlDecode()</code>	Dekodiert eine Zeichenfolge „application/x-www-form-urlencoded“.
<code>\$util.base64Encode()</code>	Codiert die Daten in eine base64-verschlüsselte Zeichenfolge.

Parameter	Beschreibung
<code>\$util.base64Decode()</code>	Decodiert die Daten einer base64-verschlüsselten Zeichenfolge.

Binäre Medientypen für WebSocket-APIs in API Gateway

API Gateway-WebSocket-APIs unterstützen derzeit keine binären Frames in Payloads von eingehenden Nachrichten. Wenn eine Client-Anwendung einen Binär-Frame sendet, weist API Gateway diesen zurück und trennt die Verbindung zum Client mit dem Code 1003.

Es gibt eine Problemumgehung für dieses Verhalten. Wenn der Client textcodierte binäre Daten (z. B. base64) als Text-Frame sendet, können Sie für die `contentHandlingStrategy`-Eigenschaft der Integration `CONVERT_TO_BINARY` festlegen, um die Nutzlast von einer base64-codierten Zeichenfolge in eine binäre Zeichenfolge zu konvertieren.

Damit für eine binäre Nutzlast in Nicht-Proxy-Integrationen eine Routenantwort zurückgegeben wird, können Sie für die `contentHandlingStrategy`-Eigenschaft der Integrationsantwort `CONVERT_TO_TEXT` festlegen, um die Nutzlast von einer binären in eine base64-codierte Zeichenfolge zu konvertieren.

Aufrufen WebSocket APIs

Nachdem Sie Ihre WebSocket API bereitgestellt haben, können Client-Anwendungen eine Verbindung zu ihr herstellen und Nachrichten an sie senden — und Ihr Back-End-Dienst kann Nachrichten an verbundene Client-Anwendungen senden:

- Sie können `wscat` damit eine Verbindung zu Ihrer WebSocket API herstellen und Nachrichten an sie senden, um das Verhalten von Clients zu simulieren. Siehe [the section called “Wird verwendet wscat, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”](#).
- Sie können die `@connections`-API aus Ihrem Backend-Service verwenden, um eine Rückrufnachricht an einen verbundenen Client zu senden, Verbindungsinformationen anzufordern oder die Clientverbindung zu trennen. Siehe [the section called “Verwenden der @connections-Befehle in Ihrem Backend-Service”](#).
- Eine Client-Anwendung kann ihre eigene WebSocket Bibliothek verwenden, um Ihre WebSocket API aufzurufen.

Wird verwendet **wscat**, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden

Das [wscat](#) Hilfsprogramm ist ein praktisches Tool zum Testen einer WebSocket API, die Sie in API Gateway erstellt und bereitgestellt haben. Sie können wscat wie folgt installieren und verwenden:

1. wscat Von <https://www.npmjs.com/package/wscat> herunterladen.
2. Installieren Sie wscat mit dem folgenden Befehl:

```
npm install -g wscat
```

3. Um eine Verbindung mit Ihrer API herzustellen, führen Sie wie im folgenden Beispiel gezeigt den Befehl wscat aus. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass die Authorization-Einstellung NONE lautet.

```
wscat -c wss://aabbccdde.execute-api.us-east-1.amazonaws.com/test/
```

Sie müssen *aabbccdde* durch die tatsächliche API-ID ersetzen, die in der API-Gateway-Konsole angezeigt oder vom AWS CLI -Befehl [create-api](#) zurückgegeben wird.

Wenn sich Ihre API in einer anderen Region als *us-east-1* befindet, müssen Sie diese außerdem durch die korrekte Region ersetzen.

4. Zum Testen Ihrer API geben Sie eine Nachricht ähnlich der folgenden ein, während Sie noch verbunden sind:

```
{"{jsonpath-expression}":"{route-key}"}
```

wo *{jsonpath-expression}* ist ein JSONPath Ausdruck und *{route-key}* ein Routenschlüssel für die API. Beispiel:

```
{"action":"action1"}  
{"message":"test response body"}
```

Weitere Informationen zu JSONPath, finden Sie unter [JSONPath](#) oder [JSONPath für Java](#).

5. Um die Verbindung mit der API zu trennen, geben Sie `ctrl-C`.

Verwenden der `@connections`-Befehle in Ihrem Backend-Service

Ihr Back-End-Dienst kann die folgenden WebSocket HTTP-Verbindungsanfragen verwenden, um eine Rückrufnachricht an einen verbundenen Client zu senden, Verbindungsinformationen abzurufen oder die Verbindung zum Client zu trennen.

Important

Da für diese Anfragen die [IAM-Autorisierung](#) erforderlich ist, müssen Sie sich mit [Signature Version 4 \(SigV4\)](#) anmelden. Dazu können Sie die API Gateway-Verwaltungs-API verwenden. Weitere Informationen finden Sie unter [ApiGatewayManagementApi](#).

Im folgenden Befehl müssen Sie ihn durch die tatsächliche API-ID `{api-id}` ersetzen, die in der API Gateway Gateway-Konsole angezeigt oder vom Befehl AWS CLI `create-api` zurückgegeben wird. Die Verbindung muss hergestellt werden, bevor Sie diesen Befehl ausführen können.

Um eine Rückrufnachricht zum Client zu senden, verwenden Sie Folgendes:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Sie können diese Anforderung mithilfe von [Postman](#) oder durch Aufruf von [awscurl](#) wie im folgenden Beispiel testen:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Sie müssen den Befehl wie im folgenden Beispiel URL-codieren:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccdde.execute-api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdFD0kwCH6w%3D
```

Zum Abrufen des neuesten Verbindungsstatus des Clients verwenden Sie:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Zum Trennen der Verbindung mit dem Client verwenden Sie:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/  
@connections/{connection_id}
```

Sie können mittels der `$context`-Variablen in Ihrer Integration eine Rückruf-URL dynamisch erstellen. Wenn Sie zum Beispiel die Lambda-Proxy-Integration mit einer Node.js-Lambda-Funktion verwenden, können Sie die URL wie folgt aufbauen und eine Nachricht an einen verbundenen Client senden:

```
import {  
  ApiGatewayManagementApiClient,  
  PostToConnectionCommand,  
} from "@aws-sdk/client-apigatewaymanagementapi";  
  
export const handler = async (event) => {  
  const domain = event.requestContext.domainName;  
  const stage = event.requestContext.stage;  
  const connectionId = event.requestContext.connectionId;  
  const callbackUrl = `https://${domain}/${stage}`;  
  const client = new ApiGatewayManagementApiClient({ endpoint: callbackUrl });  
  
  const requestParams = {  
    ConnectionId: connectionId,  
    Data: "Hello!",  
  };  
  
  const command = new PostToConnectionCommand(requestParams);  
  
  try {  
    await client.send(command);  
  } catch (error) {  
    console.log(error);  
  }  
  
  return {  
    statusCode: 200,  
  };  
};
```

Wenn Sie einen benutzerdefinierten Domainnamen für Ihre WebSocket API verwenden, entfernen Sie die `stage` Variable aus Ihrem Funktionscode.

Wenn Sie eine Rückrufnachricht senden, muss Ihre Lambda-Funktion über die Berechtigung verfügen, die API-Gateway-Management-API aufzurufen. Möglicherweise erhalten Sie eine Fehlermeldung mit dem Inhalt `GoneException`, wenn Sie eine Nachricht veröffentlichen, bevor die Verbindung hergestellt wurde oder nachdem der Client die Verbindung getrennt hat.

WebSocket-APIs für Kundenaufrufe veröffentlichen

Das einfache Erstellen und Entwickeln einer API Gateway-API macht sie nicht automatisch für Ihre Benutzer aufrufbar. Um sie aufrufbar zu machen, müssen Sie Ihre API in einer Stufe bereitstellen. Darüber hinaus können Sie die URL anpassen, die Ihre Benutzer für den Zugriff auf Ihre API verwenden. Sie können ihr eine Domäne zuweisen, die mit Ihrer Marke übereinstimmt oder einprägsamer ist als die Standard-URL für Ihre API.

In diesem Abschnitt erfahren Sie, wie Sie Ihre API bereitstellen und die URL anpassen, die Sie Benutzern für den Zugriff zur Verfügung stellen.

Note

Um die Sicherheit Ihrer API-Gateway-APIs zu erhöhen, ist die `execute-api.{region}.amazonaws.com`-Domain in der [Public Suffix List \(PSL\)](#) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host--`-Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domain-Namen für Ihre API-Gateway-APIs einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der [Set-Cookie](#)-Seite im Mozilla Developer Network.

Themen

- [Stufen für WebSocket APIs im API Gateway erstellen](#)
- [WebSocket APIs In API Gateway bereitstellen](#)
- [Sicherheitsrichtlinie für WebSocket APIs in API Gateway](#)
- [Benutzerdefinierte Domainnamen für WebSocket-APIs in API Gateway](#)

Stufen für WebSocket APIs im API Gateway erstellen

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. dev, prod, beta oder v2). API-Stufen werden durch ihre API-ID und den Stufennamen identifiziert und sind in der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

Eine Bereitstellung ist ein Snapshot Ihrer API-Konfiguration. Nachdem Sie eine API in einer Phase bereitgestellt haben, kann sie von Clients aufgerufen werden. Sie müssen eine API bereitstellen, damit Änderungen wirksam werden.

Stufenvariablen

Stufenvariablen sind Schlüssel-Wert-Paare, die Sie für eine Phase einer WebSocket API definieren können. Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung verwendet werden.

Sie können beispielsweise eine Stufenvariable definieren und dann ihren Wert als HTTP-Endpunkt für eine HTTP-Proxy-Integration festlegen. Später können Sie den Endpunkt mithilfe des zugeordneten Stufenvariablennamens referenzieren. Auf diese Weise können Sie in jeder Stufe dieselbe API-Einrichtung mit einem anderen Endpunkt verwenden. In ähnlicher Weise können Sie Stufenvariablen verwenden, um für jede Phase Ihrer API eine andere AWS Lambda Funktionsintegration anzugeben.

Note

Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Genehmigers übergeben. Weitere Informationen hierzu finden Sie unter [the section called “Antwortformat des Lambda-Genehmigers”](#).

Beispiele

Wenn Sie eine Stufenvariable zum Anpassen des HTTP-Integrationsendpunkts verwenden möchten, müssen Sie zunächst den Namen und den Wert der Stufenvariablen (z. B. url) mit dem Wert `example.com` festlegen. Richten Sie als Nächstes eine HTTP-Proxy-Integration ein. Anstatt die URL des Endpunkts einzugeben, können Sie API Gateway anweisen, den Wert der Stufenvariablen

zu verwenden, **`http://${stageVariables.url}`**. Dieser Wert weist API Gateway an, Ihre Stufenvariable `${}` zur Laufzeit abhängig von der Stufe Ihrer API zu ersetzen.

Sie können auf ähnliche Weise auf Stufenvariablen verweisen, um einen Lambda-Funktionsnamen oder einen AWS Rollen-ARN anzugeben.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Der folgende [add-permission](#)-Befehl fügt die erforderlichen Berechtigungen hinzu:

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
lambda:InvokeFunction
```

API Gateway Stufenvariablenreferenz

HTTP-Integration URIs

Sie können eine Stufenvariable als Teil einer HTTP-Integrations-URI verwenden, wie in den folgenden Beispielen gezeigt.

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Lambda-Funktionen

Sie können eine Stufenvariable anstelle eines Lambda-Funktionsnamens oder Alias verwenden, wie in den folgenden Beispielen gezeigt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/
arn:aws:lambda:<region>:<account_id>:function:<function_name>:
${stageVariables.<version_variable_name>}/invocations`

Note

Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

AWS Anmeldeinformationen für die Integration

Sie können eine Stage-Variable als Teil eines ARN mit AWS Benutzer- oder Rollenmeldedaten verwenden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

WebSocket APIs In API Gateway bereitstellen

Nachdem Sie Ihre WebSocket API erstellt haben, müssen Sie sie bereitstellen, damit Ihre Benutzer sie aufrufen können.

Zum Bereitstellen einer API erstellen Sie eine [API-Bereitstellung](#) und verknüpfen sie mit einer [Stufe](#). Jede Stufe ist ein Snapshot der API und wird für Client-Apps zum Aufrufen verfügbar gemacht.

Important

Eine API muss nach jeder Aktualisierung erneut bereitgestellt werden. Alle Änderungen außer an Stufeneinstellungen erfordern eine erneute Bereitstellung, einschließlich Änderungen an den folgenden Ressourcen:

- Routen
- Integrationen
- Genehmiger

APIs sind standardmäßig auf 10 Stufen begrenzt. Wir empfehlen, bereits verwendete Stufen für Ihre Bereitstellungen wiederzuverwenden.

Um eine bereitgestellte WebSocket API aufzurufen, sendet der Client eine Nachricht an die URL der API. Die URL wird anhand des Hostnamens und der Stufennummer der API bestimmt.

Note

API Gateway unterstützt Nutzlasten bis zu 128 KB mit einer maximalen Framegröße von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind.

Unter Verwendung des Standard-Domainnamens der API hat die URL (zum Beispiel) einer WebSocket API in einer bestimmten Phase (*{stageName}*) das folgende Format:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Um die URL der WebSocket API benutzerfreundlicher zu gestalten, können Sie einen benutzerdefinierten Domainnamen (z. B. `api.example.com`) erstellen, der den Standard-Hostnamen der API ersetzt. Der Konfigurationsprozess ist derselbe wie für REST APIs. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domänennamen”](#).

Stufen ermöglichen eine robuste Versionskontrolle Ihrer API. Sie können beispielsweise eine API für eine test- und eine prod-Stufe bereitstellen und die test-Stufe als Test-Build und die prod-Stufe als stabilen Build verwenden. Nachdem die Aktualisierungen den Test bestanden haben, können Sie die test-Stufe auf prod hochstufen. Die Hochstufung kann durch die erneute Bereitstellung der API für die prod-Stufe erfolgen. Weitere Informationen zu Stufen finden Sie unter [the section called “Einrichten einer Stufe”](#).

Themen

- [Erstellen Sie eine WebSocket API-Bereitstellung mit dem AWS CLI](#)
- [Erstellen Sie eine WebSocket API-Bereitstellung mit der API Gateway Gateway-Konsole](#)

Erstellen Sie eine WebSocket API-Bereitstellung mit dem AWS CLI

Der Befehl [create-deployment](#) erstellt eine Bereitstellung:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccdde
```

Die Ausgabe sieht wie folgt aus:

```
{
  "DeploymentId": "fedcba",
  "DeploymentStatus": "DEPLOYED",
  "CreateDate": "2018-11-15T06:49:09Z"
}
```

Die bereitgestellte API kann erst aufgerufen werden, wenn Sie die Bereitstellung einer Stufe zugeordnet haben. Sie können eine neue Stufe erstellen oder eine Stufe wiederverwenden, die Sie zuvor erstellt haben.

Der folgende [create-stage](#)-Befehl erstellt eine neue Stufe und verknüpft diese mit der Bereitstellung.

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccdde --deployment-id fedcba --stage-name test
```

Die Ausgabe sollte wie folgt aussehen:

```
{
  "StageName": "test",
  "CreateDate": "2018-11-15T06:50:28Z",
  "DeploymentId": "fedcba",
  "DefaultRouteSettings": {
    "MetricsEnabled": false,
    "ThrottlingBurstLimit": 5000,
    "DataTraceEnabled": false,
    "ThrottlingRateLimit": 10000.0
  },
  "LastUpdatedDate": "2018-11-15T06:50:28Z",
  "StageVariables": {},
  "RouteSettings": {}
}
```

Sie können eine vorhandene Phase auch wiederverwenden, indem Sie die `deploymentId` Eigenschaft der Phase mit der neu erstellten Bereitstellungs-ID (*deployment-id*) aktualisieren. Der folgende [update-stage](#)-Befehl aktualisiert die Bereitstellungs-ID der Stufe:

```
aws apigatewayv2 update-stage --region region \  
  --api-id api-id \  
  --stage-name stage-name \  
  --deployment-id deployment-id
```

Erstellen Sie eine WebSocket API-Bereitstellung mit der API Gateway Gateway-Konsole

So verwenden Sie die API Gateway Gateway-Konsole, um eine Bereitstellung für eine WebSocket API zu erstellen:

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie die API aus.
2. Klicken Sie auf Deploy API.
3. Wählen Sie die gewünschte Stufe aus der Dropdown-Liste aus oder geben Sie den Namen einer neuen Stufe ein.

Sicherheitsrichtlinie für WebSocket APIs in API Gateway

API Gateway setzt eine Sicherheitsrichtlinie von TLS_1_2 für alle WebSocket API-Endpunkte durch.

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus einer TLS-Mindestversion und Verschlüsselungsverfahren, die von Amazon API Gateway bereitgestellt wird. Das TLS-Protokoll behandelt Netzwerksicherheitsprobleme wie Manipulationen und Abhören zwischen einem Client und einem Server. Wenn Ihre Clients über die benutzerdefinierte Domäne einen TLS-Handshake mit Ihrer API ausführen, erzwingt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite-Optionen, die von Ihren Clients verwendet werden können. Diese Sicherheitsrichtlinie akzeptiert TLS 1.2- und TLS 1.3-Verkehr und weist TLS 1.0-Verkehr zurück.

Unterstützte TLS-Protokolle und Chiffren für WebSocket APIs

In der folgenden Tabelle werden die unterstützten TLS-Protokolle für beschrieben. WebSocket APIs

TLS-Protokolle	TLS_1_2-Sicherheitsrichtlinie
TLSv13.	Ja
TLSv12.	Ja

In der folgenden Tabelle werden die TLS-Verschlüsselungen beschrieben, die für die Sicherheitsrichtlinie TLS 1_2 verfügbar sind. WebSocket APIs

TLS-Verschlüsselungsverfahren	TLS_1_2-Sicherheitsrichtlinie
TLS_AES_128_GCM_SHA256	Ja
TLS_AES_256_GCM_SHA384	Ja
TLS__CHACHA20 POLY1305 SHA256	Ja
ECDHE-ECDSA- -GCM- AES128 SHA256	Ja
ECDHE-RSA- AES128 -GCM- SHA256	Ja
ECDHE-ECDSA- AES128 - SHA256	Ja
ECDHE-RSA- - AES128 SHA256	Ja
ECDHE-ECDSA- -GCM AES256 - SHA384	Ja
ECDHE-RSA- AES256 -GCM- SHA384	Ja
ECDHE-ECDSA- AES256 - SHA384	Ja
ECDHE-RSA- - AES256 SHA384	Ja
AES128-GCM- SHA256	Ja
AES128-SHA256	Ja
AES256-GCM- SHA384	Ja

TLS-Verschlüsselungsverfahren	TLS_1_2-Sicherheitsrichtlinie
AES256-SHA256	Ja

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für die gleichen Verschlüsselungsverfahren. Eine Liste der Namen der Verschlüsselungsverfahren finden Sie unter [the section called “OpenSSL- und RFC-Verschlüsselungsnamen”](#).

Informationen zu REST APIs und HTTP APIs

Weitere Informationen zu REST APIs und HTTP APIs finden Sie unter [the section called “Eine Sicherheitsrichtlinie auswählen”](#) und [the section called “Sicherheitsrichtlinie für HTTP APIs in API Gateway”](#).

Benutzerdefinierte Domainnamen für WebSocket-APIs in API Gateway

Benutzerdefinierte Domainnamen sind einfachere und intuitivere URLs, die Sie Ihren API-Benutzern zur Verfügung stellen können.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

hierzu wird *api-id* in API Gateway generiert, *region* ist die AWS-Region und *stage* wird als Stufe angegeben wenn Sie die API bereitstellen.

Der Hostname-Teil der URL (also *api-id*.execute-api.*region*.amazonaws.com) verweist auf einen API-Endpunkt. Der Standard-API-Endpunkt hat möglicherweise einen schwer zu merkenden, zufallsgenerierten Namen, der nicht besonders benutzerfreundlich ist.

Mit benutzerdefinierten Domainnamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. myservice) auswählen, um die alternative URL Ihrer API zuzuordnen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

Überlegungen

Die folgenden Überlegungen können sich auf Ihre Verwendung eines benutzerdefinierten Domainnamens auswirken.

- Wenn Sie einen benutzerdefinierten Domainnamen einer WebSocket-API zuweisen, können Sie ihn keiner REST-API oder HTTP-API zuweisen.
- Es werden ausschließlich regionale benutzerdefinierte Domainnamen unterstützt.
- Als Mindest-TLS-Version wird ausschließlich TLS 1.2 unterstützt.
- Sie müssen den Ressourceneintrag Ihres DNS-Anbieters erstellen oder aktualisieren, bevor Sie ihn dem API-Endpunkt zuordnen können. Ohne ein solches Mapping können API-Anfragen für den benutzerdefinierten Domainnamen API Gateway nicht erreichen.
- Mit einem Wildcard-Zertifikat können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das Standardkontingent zu überschreiten. Weitere Informationen finden Sie unter [the section called “Benutzerdefinierte Domainnamen mit Platzhalter”](#).

Voraussetzungen

Im Folgenden sind die Voraussetzungen für einen benutzerdefinierten Domainnamen aufgeführt.

Registrieren eines Domainnamens

Wenn Sie benutzerdefinierte Domainnamen für Ihre APIs einrichten möchten, müssen Sie eine Internetdomäne registrieren. Sie können Ihre Internetdomain mithilfe von [Amazon Route 53](#) oder mit einem Domain-Registrar Ihrer Wahl registrieren. Ihr benutzerdefinierter Domainname kann der Name einer Unter- oder Root-Domain (Zone Apex) einer registrierten Internetdomain sein.

Ihr Domainname muss der [RFC 1035](#)-Spezifikation entsprechen und darf ein Maximum von 63 Oktetten pro Label und insgesamt 255 Oktetten nicht überschreiten.

Zertifikate für benutzerdefinierte Domainnamen

Vor der Einrichtung eines benutzerdefinierten Domainnamens für eine API müssen Sie ein SSL-/TLS-Zertifikat in ACM vorbereiten. Wenn ACM in der AWS-Region nicht verfügbar ist, in der Sie Ihren benutzerdefinierten Domainnamen erstellen, müssen Sie ein Zertifikat für API Gateway in dieser Region importieren.

Wenn Sie ein SSL-/TLS-Zertifikat importieren möchten, müssen Sie den PEM-formatierten Hauptteil des SSL-/TLS-Zertifikats, den persönlichen Schlüssel und die Zertifikatkette für den benutzerdefinierten Domainnamen bereitstellen.

Ein in ACM gespeichertes Zertifikat wird durch seinen ARN angegeben. Wenn Sie Zertifikate verwenden, die von ACM ausgestellt werden, brauchen sich keine Gedanken über vertrauliche Zertifikatdetails wie private Schlüssel zu machen. Wenn Sie also ein von AWS verwaltetes Zertifikat für einen Domainnamen verwenden möchten, müssen Sie lediglich auf dessen ARN verweisen.

Wenn Ihre Anwendung, manchmal auch als Zertifikats-Pinning bekannt, verwendet, um ein ACM-Zertifikat zu pinnen, kann die Anwendung möglicherweise keine Verbindung zu Ihrer Domain herstellen, nachdem AWS das Zertifikat erneuert hat. Weitere Informationen finden Sie unter [Probleme beim Zertifikats-Pinning](#) im AWS Certificate Manager-Benutzerhandbuch.

Benutzerdefinierte Domainnamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domainnamen können Sie eine nahezu unendliche Anzahl von Domainnamen unterstützen, ohne das [Standardkontingent](#) zu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domainnamen geben, *customername*.api.example.com.

Um einen benutzerdefinierten Platzhalterdomainnamen zu erstellen, geben Sie einen Platzhalter (*) als erste Subdomain einer benutzerdefinierten Domain an, die alle möglichen Subdomains einer Root-Domain darstellt.

Der benutzerdefinierte Domainnamen mit Platzhalter *.example.com führt beispielsweise zu Unterdomains wie a.example.com, b.example.com und c.example.com, die alle zur gleichen Domain weiterleiten.

Benutzerdefinierte Domainnamen mit Platzhaltern unterstützen andere Konfigurationen als die benutzerdefinierten Standarddomainnamen von API Gateway. Beispielsweise können Sie in einem einzelnen AWS-Konto *.example.com und a.example.com so konfigurieren, dass sie sich anders verhalten.

Sie können die Kontextvariablen `$context.domainName` und `$context.domainPrefix` verwenden, um den Domainnamen zu bestimmen, mit dem ein Client Ihre API aufruft. Weitere Informationen zu Kontextvariablen finden Sie unter [Variablen für Datentransformationen für API Gateway](#).

Um einen benutzerdefinierten Domainnamen mit Platzhalter zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat angeben, das mithilfe der DNS- oder der E-Mail-Validierungsmethode validiert wurde.

 Note

Sie können keinen benutzerdefinierten Domainnamen mit Platzhalter erstellen, wenn ein anderes AWS-Konto einen benutzerdefinierten Domainnamen erstellt hat, der mit dem benutzerdefinierten Domainnamen mit Platzhalter in Konflikt steht. Wurde `a.example.com` beispielsweise von Konto A erstellt, dann kann Konto B als benutzerdefinierten Domainnamen mit Platzhalter nicht `*.example.com` erstellen.

Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das [AWS Supportcenter](#) wenden, um eine Ausnahme anzufordern.

Die nächsten Schritte für benutzerdefinierte Domainnamen

Verwenden Sie die Dokumentation aus dem REST-API-Abschnitt des API-Gateway-Entwicklerhandbuchs für die Einrichtung eines benutzerdefinierten Domainnamens für eine HTTP-API.

Geben Sie zunächst ein Zertifikat für Ihren benutzerdefinierten Domainnamen an. Weitere Informationen finden Sie unter [the section called “Bereiten Sie Zertifikate vor in AWS Certificate Manager”](#). Dann erstellen Sie einen regionalen benutzerdefinierten Domainnamen. Weitere Informationen finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

API-Stufen einem benutzerdefinierten Domainnamen für WebSocket-APIs zuweisen

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Nachdem Sie einen Domain-Namen erstellt und DNS-Einträge konfiguriert haben, verwenden Sie API-Mappings, um Datenverkehr über Ihren benutzerdefinierten Domain-Namen an Ihre APIs zu senden.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise die `production`-Phase einer API in `wss://api.example.com/orders` abbilden.

Bevor Sie eine API-Zuweisung erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten

Domain-Namens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

Einschränkungen

- In einer API-Zuweisung müssen sich der benutzerdefinierte Domainname und die zugeordneten APIs im selben AWS-Konto befinden.
- API-Zuweisungen dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: \$ - _ . + ! * ' ().
- Die maximale Länge für den Pfad in einer API-Zuweisung beträgt 300 Zeichen.
- Sie können WebSocket-APIs nicht demselben benutzerdefinierten Domainnamen wie dem einer HTTP-API oder REST-API zuweisen.
- Wenn Sie mehrstufige API-Zuweisungen erstellen, konvertiert API Gateway alle Header-Namen in Kleinbuchstaben.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter [the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens”](#).

AWS-Managementkonsole

So erstellen Sie eine API-Zuweisung

1. Melden Sie sich bei der API-Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie benutzerdefinierte Domain-Namen aus.
3. Wählen Sie einen benutzerdefinierten Domain-Namen aus, den Sie bereits erstellt haben.
4. Wählen Sie API-Mappings aus.
5. Wählen Sie API-Zuweisungen konfigurieren aus.
6. Wählen Sie Neue Zuweisung hinzufügen aus.
7. Geben Sie eine API, eine Phase und optional einen Pfad ein.
8. Wählen Sie Speichern aus.

AWS CLI

Der folgende [create-api-mapping](#)-Befehl erstellt eine API-Zuweisung. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1` an die angegebene API und Phase.

```
aws apigatewayv2 create-api-mapping \  
  --domain-name api.example.com \  
  --api-mapping-key v1 \  
  --api-id a1b2c3d4 \  
  --stage test
```

CloudFormation

Mit dem folgenden CloudFormation-Beispiel wird ein API-Mapping erstellt.

```
MyApiMapping:  
  Type: 'AWS::ApiGatewayV2::ApiMapping'  
  Properties:  
    DomainName: api.example.com  
    ApiMappingKey: 'v1'  
    ApiId: !Ref MyApi  
    Stage: !Ref MyStage
```

IP-Adresstypen für benutzerdefinierte Domainnamen für WebSocket-APIs

Wenn Sie einen benutzerdefinierten Domainnamen erstellen, geben Sie den Typ der IP-Adressen an, die Ihre Domain aufrufen können. Sie können IPv4 auswählen, um IPv4-Adressen für den Aufruf Ihrer Domain zuzulassen, oder Dualstack, um sowohl IPv4- als auch IPv6-Adressen für den Aufruf zuzulassen. Wir empfehlen, den IP-Adresstyp auf Dualstack zu setzen, um Engpässe im IP-Adressraum zu vermeiden und Ihre Sicherheitsposition zu verbessern. Weitere Informationen zu den Vorteilen eines Dualstack-IP-Adresstyps finden Sie unter [IPv6 in AWS](#).

Überlegungen zu IP-Adresstypen

Die folgenden Überlegungen können Ihre Verwendung von IP-Adresstypen beeinflussen.

- Der Standard-IP-Adresstyp für benutzerdefinierte Domainnamen in API Gateway ist IPv4.

- Ihr benutzerdefinierter Domainname muss nicht denselben IP-Adresstyp für alle darauf zugewiesenen APIs haben. Wenn Sie Ihren Standard-API-Endpunkt deaktivieren, kann dies die Art und Weise beeinflussen, wie Aufrufer Ihre API aufrufen können.

Ändern des IP-Adresstyps eines benutzerdefinierten Domainnamens

Sie können den IP-Adresstyp ändern, indem Sie die Endpunktkonfiguration des Domainnamens aktualisieren. Dies kann über die AWS-Managementkonsole, die AWS CLI, CloudFormation oder ein AWS SDK erfolgen.

AWS-Managementkonsole

So ändern Sie den IP-Adresstyp eines benutzerdefinierten Domainnamens

1. Melden Sie sich bei der API Gateway-Konsole unter <https://console.aws.amazon.com/apigateway> an.
2. Wählen Sie einen öffentlichen benutzerdefinierten Domainnamen aus.
3. Wählen Sie Endpunktkonfiguration aus.
4. Wählen Sie unter „IP-Adresstyp“ entweder IPv4 oder Dualstack aus.
5. Wählen Sie Speichern aus.

AWS CLI

Der folgende [update-domain-name](#)-Befehl aktualisiert eine API, sodass sie den IP-Adresstyp „Dualstack“ aufweist:

```
aws apigatewayv2 update-domain-name \
  --domain-name dualstack.example.com \
  --domain-name-configurations CertificateArn=arn:aws:acm:us-
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

Die Ausgabe sieht wie folgt aus:

```
{
  "ApiMappingSelectionExpression": "$request.basepath",
  "DomainName": "dualstack.example.com",
  "DomainNameConfigurations": [
    {
```

```
    "ApiGatewayDomainName": "d-abcd1234.execute-api.us-  
east-1.amazonaws.com",  
    "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/  
abcd1234-5678-abc",  
    "DomainNameStatus": "AVAILABLE",  
    "EndpointType": "REGIONAL",  
    "HostedZoneId": "Z3LQWSYCGH4ADY",  
    "SecurityPolicy": "TLS_1_2",  
    "IpAddressType": "dualstack"  
  }  
],  
  "Tags": {}  
}
```

Deaktivieren Sie den Standardendpunkt für WebSocket APIs

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domännennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt. Wenn Sie den Standardendpunkt deaktivieren, wirkt sich dies auf alle Stufen einer API aus.

Das folgende Verfahren zeigt, wie Sie den Standardendpunkt für eine WebSocket API deaktivieren.

AWS-Managementkonsole

1. Melden Sie sich bei der API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigatewayan>.
2. Wählen Sie eine API aus. WebSocket
3. Wählen Sie API-Einstellungen aus.
4. Klicken Sie unter API-Details auf Bearbeiten.
5. Wählen Sie unter Standardendpunkt die Option Inaktiv aus.
6. Wählen Sie Änderungen speichern aus.
7. Klicken Sie im Navigationsbereich auf Routen.
8. Klicken Sie auf Bereitstellen und stellen Sie Ihre API erneut bereit oder erstellen Sie eine neue Stufe, in der die Änderung wirksam werden soll.

AWS CLI

Der folgende [update-api-Befehl deaktiviert den Standardendpunkt für eine API](#): WebSocket

```
aws apigatewayv2 update-api \  
  --api-id abcdef123 \  
  --disable-execute-api-endpoint
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird.

Der folgende AWS CLI Befehl erstellt eine Bereitstellung.

```
aws apigatewayv2 create-deployment \  
  --api-id abcdef123 \  
  --stage-name dev
```

Schützen Sie Ihr WebSocket APIs internes API Gateway

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, wie böswillige Benutzer oder Spitzen im Datenverkehr. Sie können Ihre API mit Strategien wie dem Generieren von SSL-Zertifikaten oder dem Festlegen von Drosselungszielen schützen. Weitere Informationen über das Generieren von Zertifikaten finden Sie unter [the section called "Clientzertifikate"](#). Der Rest dieses Abschnitts behandelt das Festlegen von Drosselungszielen.

Sie können die Drosselung für Sie konfigurieren, um sie davor APIs zu schützen, von zu vielen Anfragen überfordert zu werden. Drosselungen werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere untersucht API Gateway die Rate und die Anzahl der eingereichten Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise 429 Too Many Requests Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Zielgrenzwerte für einzelne API-Phasen oder -Routen festlegen, um die Gesamtleistung APIs in Ihrem Konto zu verbessern.

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) APIs innerhalb eines AWS Kontos pro Region. Außerdem wird der Burst (d. h. die maximale Bucket-Größe) für alle APIs innerhalb eines AWS Kontos pro Region begrenzt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von 429 Too Many Requests-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter [Kontingente und wichtige Hinweise](#).

Die Limits pro Konto gelten für alle APIs Konten in einer bestimmten Region. Das Ratenlimit auf Kontoebene kann auf Anfrage erhöht werden. Höhere Limits sind möglich, was kürzere Timeouts und geringere Payloads zur Folge hat. APIs Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das [AWS Supportcenter](#). Weitere Informationen finden Sie unter [Kontingente und wichtige Hinweise](#). Beachten Sie, dass diese Limits nicht höher als die Drosselungsgrenzen sein können. AWS

Drosselung auf Routenebene

Sie können die Drosselung auf Routenebene festlegen, um die Anforderungs-Drosselungslimits auf Kontoebene für eine bestimmte Stufe oder für individuelle Routen in Ihrer API zu überschreiben. Die Standardgrenzen für die Routendrosselung können die Ratenlimits auf Kontoebene nicht überschreiten.

Sie können die Drosselung auf Routenebene konfigurieren, indem Sie die verwerde AWS CLI. Der folgende [update-stage](#)-Befehl konfiguriert die benutzerdefinierte Drosselung für die angegebene Stufe und Route einer API.

```
aws apigatewayv2 update-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-settings '{"messages":  
{"ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000}}'
```

WebSocket-APIs in API Gateway überwachen

Sie können CloudWatch-Metriken und CloudWatch Logs zur Überwachung von WebSocket-APIs verwenden. Durch die Kombination von Protokollen und Metriken können Sie Fehler protokollieren und die Leistung Ihrer API überwachen.

Note

API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:

- 413 Request Entity Too Large-Fehler
- Übermäßige 429 Too Many Requests-Fehler
- Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
- Fehler der Serie 500, die durch interne Ausfälle verursacht werden

Themen

- [Überwachen Sie die WebSocket API-Ausführung mit CloudWatch Metriken](#)
- [Konfigurieren Sie die Protokollierung für WebSocket APIs im API Gateway](#)

Überwachen Sie die WebSocket API-Ausführung mit CloudWatch Metriken

Sie können [CloudWatchAmazon-Metriken](#) zur Überwachung verwenden WebSocket APIs. Die Konfiguration ähnelt der für REST verwendeten APIs. Weitere Informationen finden Sie unter [Überwachen Sie die REST-API-Ausführung mit CloudWatch Amazon-Metriken](#).

Die folgenden Metriken werden unterstützt für WebSocket APIs:

Metrik	Description
ConnectCount	Die Anzahl der Nachrichten, die an die \$connect-Routenintegration gesendet werden.

Metrik	Description
MessageCount	Die Anzahl der Nachrichten, die entweder vom oder an den Client an die WebSocket API gesendet wurden.
IntegrationError	Die Anzahl der Anforderungen, die eine 4XX/5XX-Antwort von der Integration zurückgeben.
ClientError	Die Anzahl der Anfragen, auf die eine 4XX-Antwort von API Gateway zurückgegeben wird, bevor die Integration aufgerufen wird.
ExecutionError	Fehler beim Aufrufen der Integration.
IntegrationLatency	Der Zeitunterschied zwischen dem Senden der Anfragen durch API Gateway an die Integration und dem Eingang der Antwort der Integration bei API Gateway. Wird für Rückrufe und Pseudo-Integrationen ausgeblendet.

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

Dimension	Description
Apild	Filtert API Gateway-Metriken für eine API mit der angegebenen API-ID.
Apild, Phase	Filtert API Gateway-Metriken für eine API-Stufe mit der angegebenen API-ID und Stufen-ID.
Apild, Methode, Ressource, Phase	Filtert API-Gateway-Metriken für eine API-Methode mit der angegebenen API-ID, Stufen-ID, dem Ressourcenpfad und der Routen-ID. API Gateway sendet diese Metriken nur, wenn Sie detaillierte CloudWatch Metriken explizit aktiviert haben. Sie können dies tun, indem Sie die UpdateStageAktion der API Gateway V2 REST-API aufrufen, auf die die <code>detailedMetricsEnabled</code> Eigenschaft aktualisiert <code>true</code> werden soll. Alternativ können Sie den AWS CLI Befehl update-stage aufrufen, um die <code>DetailedMetricsEnabled</code> Eigenschaft auf <code>true</code> zu aktualisieren. Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen Gebühren belastet. Preisinfo

Dimension	Description
	Informationen finden Sie unter CloudWatchAmazon-Preise .

Konfigurieren Sie die Protokollierung für WebSocket APIs im API Gateway

Sie können die Protokollierung aktivieren, um CloudWatch Protokolle in Logs zu schreiben. Es gibt zwei Arten der API-Anmeldung CloudWatch: Ausführungsprotokollierung und Zugriffsprotokollierung. Bei der Ausführungsprotokollierung verwaltet API Gateway die CloudWatch Protokolle. Der Prozess umfasst das Erstellen von Protokollgruppen und -Streams sowie die Meldung der Anforderungen und Antworten jedes Aufrufers an Protokoll-Streams.

Zur Verbesserung Ihrer Sicherheitslage empfehlen wir Ihnen, die Ausführungsprotokollierung auf der ERROR- oder INFO-Ebene zu verwenden. Dies kann erforderlich sein, um verschiedene Compliance-Rahmenwerke einzuhalten. Weitere Informationen finden Sie unter [Amazon-API-Gateway-Steuerelemente](#) im Benutzerhandbuch für AWS Security Hub .

In der Zugriffsprotokollierung protokollieren Sie, als API-Entwickler, wer auf Ihre API zugegriffen hat und wie der Aufrufer auf die API zugegriffen hat. Erstellen Sie eine eigene Protokollgruppe oder wählen Sie eine vorhandene Protokollgruppe aus, die von API Gateway verwaltet werden kann. Um die Zugriffsdetails anzugeben, wählen Sie `$context`-Variablen (ausgedrückt in einem Format Ihrer Wahl) und eine Protokollgruppe als Ziel aus.

Anweisungen zum Einrichten der CloudWatch Protokollierung finden Sie unter [the section called “CloudWatch API-Protokollierung mit der API Gateway Gateway-Konsole einrichten”](#).

Wenn Sie das Log Format (Protokollformat) angeben, können Sie wählen, welche Kontextvariablen protokolliert werden sollen. Die folgenden Variablen werden unterstützt.

Parameter	Beschreibung
<code>\$context.apiId</code>	Die ID, die API Gateway Ihrer API zuweist.
<code>\$context.authorize.error</code>	Die Autorisierungsfehlermeldung.
<code>\$context.authorize.latency</code>	Die Autorisierungslatenz in ms

Parameter	Beschreibung
<code>\$context.authorize.status</code>	Der Statuscode, der von einem Autorisierungsversuch zurückgegeben wurde.
<code>\$context.authorizer.error</code>	Die von einem Genehmiger zurückgegebene Fehlermeldung.
<code>\$context.authorizer.integrationLatency</code>	Die Lambda-Genehmiger-Latenzzeit in ms.
<code>\$context.authorizer.integrationStatus</code>	Der von einem Lambda-Genehmiger zurückgegebene Statuscode.
<code>\$context.authorizer.latency</code>	Der Genehmiger-Latenz in ms.
<code>\$context.authorizer.requestId</code>	Die Anforderungs-ID des AWS Endpunkts.
<code>\$context.authorizer.status</code>	Der von einem Genehmiger zurückgegebene Statuscode.
<code>\$context.authorizer.principalId</code>	Die ID des Prinzipalbenutzers, die mit dem vom Client gesendeten und von einer Lambda-Genehmiger-Lambda-Funktion des API Gateways zurückgegebenen Token verknüpft ist. (Ein Lambda-Genehmiger wurde früher als benutzerdefinierter Genehmiger bezeichnet).

Parameter	Beschreibung
<code>\$context.authorizer.</code> <i>property</i>	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der context-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende context-Zuweisung zurück: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Dann gibt der Aufruf von <code>\$context.authorizer.key</code> die Zeichenfolge "value" zurück, der Aufruf von <code>\$context.authorizer.numKey</code> gibt die Zeichenfolge "1" zurück und der Aufruf von <code>\$context.authorizer.boolKey</code> gibt den Wert "true" zurück.
<code>\$context.authenticate.error</code>	Die von einem Authentifizierungsversuch zurückgegebene Fehlermeldung.
<code>\$context.authenticate.latency</code>	Die Authentifizierungslatenz in ms
<code>\$context.authenticate.status</code>	Der Statuscode, der von einem Authentifizierungsversuch zurückgegeben wurde.
<code>\$context.connectedAt</code>	Die Epoch -formatierte Verbindungszeit.

Parameter	Beschreibung
<code>\$context.connectionId</code>	Eine eindeutige ID für die Verbindung, die für einen Rückruf an den Client verwendet werden kann.
<code>\$context.domainName</code>	Ein Domainname für die WebSocket API. Dies kann für einen Rückruf an den Client (anstelle eines hardcodierten Wertes) verwendet werden.
<code>\$context.error.message</code>	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält.
<code>\$context.error.messageString</code>	Der Wert von <code>\$context.error.message</code> in Anführungszeichen, d. h. " <code>\$context.error.message</code> ".
<code>\$context.error.responseType</code>	Der Fehler-Antworttyp.
<code>\$context.error.validationErrorString</code>	Eine Zeichenfolge, die eine detaillierte Validierungsfehlermeldung enthält.
<code>\$context.eventType</code>	Der Ereignistyp: <code>CONNECT</code> , <code>MESSAGE</code> oder <code>DISCONNECT</code> .
<code>\$context.extendedRequestId</code>	Äquivalent mit <code>\$context.requestId</code> .
<code>\$context.identity.accountId</code>	Die der Anfrage zugeordnete AWS Konto-ID.
<code>\$context.identity.apiKey</code>	Der API-Besitzerschlüssel, der der schlüsselfähigen API-Anforderung zugewiesen ist.
<code>\$context.identity.apiKeyId</code>	Die API-Schlüssel-ID, die der schlüsselfähigen API-Anforderung zugewiesen ist.

Parameter	Beschreibung
<code>\$context.identity.caller</code>	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.cognitoAuthenticationProvider</code>	Eine durch Komma getrennte Liste aller Amazon-Cognito-Authentifizierungsanbieter, die vom anfordernden Aufrufer verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, <code>cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Weitere Informationen zu verfügbaren Amazon-Cognito-Authentifizierungsanbietern finden Sie unter Verbundidentitäten verwenden im Amazon-Cognito-Entwicklerhandbuch.
<code>\$context.identity.cognitoAuthenticationType</code>	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind <code>authenticated</code> für authentifizierte Identitäten und <code>unauthenticated</code> für nicht authentifizierte Identitäten.

Parameter	Beschreibung
<code>\$context.identity.cognitoId</code> <code>entityId</code>	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.cognitoId</code> <code>entityPoolId</code>	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
<code>\$context.identity.principalOrgId</code>	Die AWS -Organisations-ID . Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.sourceIp</code>	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
<code>\$context.identity.user</code>	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
<code>\$context.identity.userAgent</code>	Der Benutzeragent des API-Aufrufers.
<code>\$context.identity.userArn</code>	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung.
<code>\$context.integration.error</code>	Die von einer Integration zurückgegebene Fehlermeldung.
<code>\$context.integration.integrationStatus</code>	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.

Parameter	Beschreibung
<code>\$context.integration.latency</code>	Die Integrationslatenz in Millisekunden. Äquivalent mit <code>\$context.integrationLatency</code> .
<code>\$context.integration.requestId</code>	Die AWS Anforderungs-ID des Endpunkts . Äquivalent mit <code>\$context.awsEndpointRequestId</code> .
<code>\$context.integration.status</code>	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird. Äquivalent mit <code>\$context.integrationStatus</code> .
<code>\$context.integrationLatency</code>	Die Integrationslatenz in ms, nur für die Zugriffsprotokollierung verfügbar.
<code>\$context.messageId</code>	Eine eindeutige serverseitige ID für eine Nachricht. Nur verfügbar, wenn der <code>\$context.eventType</code> MESSAGE ist.
<code>\$context.requestId</code>	Entspricht <code>\$context.extendedRequestId</code> .
<code>\$context.requestTime</code>	Die Anforderungszeit im CLF -Format (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	Die Anforderungszeit im Epoch -Format in Millisekunden.
<code>\$context.routeKey</code>	Der ausgewählte Routenschlüssel.
<code>\$context.stage</code>	Die Bereitstellungsstufe des API-Aufrufs (z. B. "beta" oder "prod").
<code>\$context.status</code>	Der Antwortstatus.

Parameter	Beschreibung
<code>\$context.waf.error</code>	Die Fehlermeldung wurde von zurückgegeben AWS WAF.
<code>\$context.waf.latency</code>	Die AWS WAF Latenz in ms.
<code>\$context.waf.status</code>	Der Statuscode wurde von zurückgegeben AWS WAF.

Beispiele für einige häufig verwendete Zugriffsprotokollformate werden in der API Gateway-Konsole dargestellt und wie folgt aufgeführt.

- CLF ([Common Log Format](#)):

```
$context.identity.sourceIp $context.identity.caller \  
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey  
$context.connectionId" \  
$context.status $context.requestId
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- JSON:

```
{  
  "requestId": "$context.requestId", \  
  "ip": "$context.identity.sourceIp", \  
  "caller": "$context.identity.caller", \  
  "user": "$context.identity.user", \  
  "requestTime": "$context.requestTime", \  
  "eventType": "$context.eventType", \  
  "routeKey": "$context.routeKey", \  
  "status": "$context.status", \  
  "connectionId": "$context.connectionId"  
}
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- XML:

```
<request id="$context.requestId"> \  
  <ip>$context.identity.sourceIp</ip> \  
  <caller>$context.identity.caller</caller> \  
  <user>$context.identity.user</user> \  
  <requestTime>$context.requestTime</requestTime> \  
  <eventType>$context.eventType</eventType> \  
  <routeKey>$context.routeKey</routeKey> \  
  <status>$context.status</status> \  
  <connectionId>$context.connectionId</connectionId> \  
</request>
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.identity.caller, \  
$context.identity.user,$context.requestTime,$context.eventType, \  
$context.routeKey,$context.connectionId,$context.status, \  
$context.requestId
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

Referenz zu API Gateway Amazon-Ressourcenname (ARN)

In den folgenden Tabellen sind die Amazon-Ressourcenamen (ARNs) für API Gateway Gateway-Ressourcen aufgeführt. Weitere Informationen zur Verwendung von ARNs AWS Identity and Access Management Richtlinien finden Sie unter [Funktionsweise von Amazon API Gateway mit IAM](#) und [Zugriffskontrolle für eine API mit IAM-Berechtigungen](#).

HTTP-API und WebSocket API-Ressourcen

Ressource	ARN
AccessLogSettings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> / stages/ <i>stage-name</i> /accesslo gsettings
Api	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i>
Apis	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis
Authorizer	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /authoriz ers/ <i>id</i>
Genehmiger	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /authoriz ers
Cors	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /cors
Bereitstellung	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /deployme nts/ <i>id</i>

Ressource	ARN
Bereitstellungen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /deployments
ExportedAPI	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /exports/ <i>specification</i>
Integration	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /integrations/ <i>integration-id</i>
Integrationen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /integrations
IntegrationResponse	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /integrationresponses/ <i>integration-response</i>
IntegrationResponses	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /integrationresponses
Modell	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /models/ <i>id</i>
Modelle	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /models
ModelTemplate	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /models/ <i>id</i> /template
Route	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i>

Ressource	ARN
Routen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /routes
RouteRequestParameter	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> / requestparameters/ <i>key</i>
RouteResponse	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> / routeresponses/ <i>id</i>
RouteResponses	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> / routeresponses
RouteSettings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> / stages/ <i>stage-name</i> /routeset tings/ <i>route-key</i>
Stage	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> / stages/ <i>stage-name</i>
Stufen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /stages
VpcLink	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/vpclinks/ <i>vpclink-id</i>
VpcLinks	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/vpclinks

REST-API-Ressourcen

Ressource	ARN
Account	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/account
ApiKey	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apikeys/ <i>id</i>
ApiKeys	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apikeys
Authorizer	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / authorizers/ <i>id</i>
Genehmiger	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / authorizers
ClientCertificate	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/clientcertifica tes/ <i>id</i>
ClientCertificates	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/clientcertificates
Bereitstellung	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / deployments/ <i>id</i>
Bereitstellungen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / deployments

Ressource	ARN
DocumentationPart	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / documentation/parts/ <i>id</i>
DocumentationParts	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / documentation/parts
DocumentationVersion	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / documentation/versions/ <i>version</i>
DocumentationVersions	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / documentation/versions
GatewayResponse	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / gatewayresponses/ <i>response-type</i>
GatewayResponses	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / gatewayresponses
Integration	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>resource-id</i> /methods/ <i>http-method</i> /integration
IntegrationResponse	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>resource-id</i> /methods/ <i>http-method</i> /integration/respo nses/ <i>status-code</i>

Ressource	ARN
Methode	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>resource-id</i> /methods/ <i>http-method</i>
MethodResponse	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>resource-id</i> /methods/ <i>http-method</i> /responses/ <i>status-code</i>
Modell	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / models/ <i>model-name</i>
Modelle	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / models
RequestValidator	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / requestvalidators/ <i>id</i>
RequestValidators	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / requestvalidators
Ressource	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>id</i>
Ressourcen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources

Ressource	ARN
RestApi	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i>
RestApis	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis
Stage	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / stages/ <i>stage-name</i>
Stufen	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / stages
Tags (Markierungen)	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/tags/ <i>url-encoded-resource-arn</i>
Vorlage	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/models / <i>model-name</i> /template
UsagePlan	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i>
UsagePlans	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/usageplans
UsagePlanKey	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i> /keys/ <i>id</i>
UsagePlanKeys	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i> /keys

Ressource	ARN
VpcLink	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/vpclinks/ <i>vpclink-id</i>
VpcLinks	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/vpclinks

Ressourcen für Domainnamen

Ressource	ARN
ApiMapping	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /apimappings/ <i>id</i>
ApiMappings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /apimappings
BasePathMapping	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /basepathmappings/ <i>basepath</i>
BasePathMappings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /basepathmappings
DomainName	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-name</i>
DomainNameAccessAssociation	arn: <i>partition</i> :apigatew ay: <i>region</i> : <i>account-id</i> :/domainnameaccessassociations/domainname/ <i>domain-name</i> /vpcesource/ <i>vpce-source-id</i>

Ressource	ARN
DomainNameAccessAssociations	arn: <i>partition</i> :apigateway: <i>region</i> : <i>account-id</i> :/domainnameaccessassociations
DomainNames	arn: <i>partition</i> :apigateway: <i>region</i> ::/domainnames
RoutingRule	arn: <i>partition</i> :apigateway: <i>region</i> : <i>account-id</i> :/domainnames/ <i>domain-name</i> /routingrules/ <i>routing-rule-id</i>

execute-api(HTTP APIs, WebSocket APIs, und REST APIs)

Ressource	ARN
WebSocket API-Endpoint	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> / <i>stage</i> / <i>route-key</i>
HTTP-API- und REST-API-Endpoint *	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> / <i>stage</i> / <i>http-method</i> / <i>resource-path</i>
Lambda-Genehmiger **	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> /authorizers/ <i>authorizer-id</i>

* Der ARN für den `$default` Routenendpunkt für HTTP APIs lautet `arn:partition:execute-api:region:account-id:api-id/*/$default`.

****** Dieser ARN ist nur für die Einrichtung der SourceArn-Bedingung in der [Ressourcenrichtlinie](#) für eine Lambda-Genehmiger-Funktion anwendbar. Ein Beispiel finden Sie unter [the section called “Erstellen eines Lambda-Genehmigers”](#).

OpenAPI-Erweiterungen für API Gateway

Die API Gateway Gateway-Erweiterungen unterstützen die AWS-spezifische Autorisierung und die API Gateway Gateway-spezifischen API-Integrationen für REST und HTTP. APIs APIs In diesem Abschnitt beschreiben wir die API Gateway-Erweiterungen der OpenAPI-Spezifikation.

Tip

Um zu verstehen, wie die API Gateway-Erweiterungen in einer Anwendung verwendet werden, können Sie die API Gateway-Konsole verwenden, um eine REST-API oder HTTP-API zu erstellen und diese in eine OpenAPI-Definitionsdatei zu exportieren. Weitere Informationen zum Exportieren einer API finden Sie unter [REST-API von API Gateway importieren](#) und [Exportieren Sie HTTP APIs aus dem API Gateway](#).

Themen

- [x-amazon-apigateway-any-method-Objekt](#)
- [x-amazon-apigateway-cors Objekt](#)
- [x-amazon-apigateway-apiEigenschaft -key-source](#)
- [x-amazon-apigateway-auth Objekt](#)
- [x-amazon-apigateway-authorizer Objekt](#)
- [x-amazon-apigateway-authtype Eigentum](#)
- [x-amazon-apigateway-binaryEigenschaft -media-types](#)
- [x-amazon-apigateway-documentation Objekt](#)
- [x-amazon-apigateway-endpoint-Zugriffsmodus](#)
- [x-amazon-apigateway-endpoint-Konfigurationsobjekt](#)
- [x-amazon-apigateway-gatewayObjekt -Responses](#)
- [x-amazon-apigateway-gateway-Responses.GatewayResponse-Objekt](#)
- [x-amazon-apigateway-gateway-Responses.ResponseParameters-Objekt](#)
- [x-amazon-apigateway-gateway-Responses.ResponseTemplates-Objekt](#)
- [x-amazon-apigateway-importexport-Version](#)

- [x-amazon-apigateway-integration Objekt](#)
- [x-amazon-apigateway-integrations Objekt](#)
- [x-amazon-apigateway-integration.requestTemplates-Objekt](#)
- [x-amazon-apigateway-integration.RequestParameters Objekt](#)
- [x-amazon-apigateway-integration.responses-Objekt](#)
- [x-amazon-apigateway-integration.response-Objekt](#)
- [x-amazon-apigateway-integration.responseTemplates-Objekt](#)
- [x-amazon-apigateway-integration.responseParameters-Objekt](#)
- [x-amazon-apigateway-integration.tlsConfig-Objekt](#)
- [x-amazon-apigateway-minimum-Kompressionsgröße](#)
- [x-amazon-apigateway-policy](#)
- [x-amazon-apigateway-request-validator-Eigenschaft](#)
- [x-amazon-apigateway-request-validators-Objekt](#)
- [x-amazon-apigateway-request-Validators.RequestValidator-Objekt](#)
- [x-amazon-apigateway-security-Richtlinie](#)
- [x-amazon-apigateway-tagEigenschaft -value](#)

x-amazon-apigateway-any-method-Objekt

Gibt das [OpenAPI-Operations-Objekt](#) für die Catch-all-Methode ANY von API Gateways in einem [OpenAPI-Pfad-Element-Objekt](#) an. Dieses Objekt kann neben anderen Operation-Objekten bestehen und fängt sämtliche HTTP-Methoden ab, die nicht explizit deklariert wurden.

Die folgende Tabelle listet die durch API Gateway erweiterten Eigenschaften auf. Andere OpenAPI-Operation-Eigenschaften finden Sie in der OpenAPI-Spezifikation.

Eigenschaftensname	Typ	Beschreibung
<code>isDefaultRoute</code>	Boolean	Gibt an, ob eine Route die \$default-Route ist. Wird nur für HTTP APIs unterstützt. Weitere Informationen hierzu

Eigenschaftename	Typ	Beschreibung
		finden Sie unter Routen für HTTP APIs in API Gateway erstellen .
x-amazon-apigateway-integration	x-amazon-apigateway-integration Objekt	Gibt die Integration der Methode mit dem Backend an. Hierbei handelt es sich um eine erweiterte Eigenschaft des Objekts OpenAPI-Operation . Die Integration kann vom Typ AWS, AWS_PROXY, HTTP, HTTP_PROXY oder MOCK sein.

x-amazon-apigateway-any-Methodenbeispiele

Im folgenden Beispiel wird die ANY-Methode auf einer Proxy-Ressource {proxy+} mit einer Lambda-Funktion TestSimpleProxy integriert.

```
"/{proxy+}": {
  "x-amazon-apigateway-any-method": {
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "proxy",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {},
    "x-amazon-apigateway-integration": {
      "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:TestSimpleProxy/invocations",
      "httpMethod": "POST",
      "type": "aws_proxy"
    }
  }
}
```

```
}
```

Das folgende Beispiel erstellt eine `$default`-Route für eine HTTP-API, die sich in eine Lambda-Funktion (HelloWorld) integriert.

```
"/$default": {  
  "x-amazon-apigateway-any-method": {  
    "isDefaultRoute": true,  
    "x-amazon-apigateway-integration": {  
      "type": "AWS_PROXY",  
      "httpMethod": "POST",  
      "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",  
      "timeoutInMillis": 1000,  
      "connectionType": "INTERNET",  
      "payloadFormatVersion": 1.0  
    }  
  }  
}
```

x-amazon-apigateway-cors Objekt

Gibt die CORS-Konfiguration (Cross-Origin Resource Sharing) für eine HTTP-API an. Die Erweiterung gilt für die OpenAPI-Struktur auf Stammebene. Weitere Informationen hierzu finden Sie unter [CORS für HTTP APIs im API Gateway konfigurieren](#).

Eigenschaftename	Typ	Beschreibung
<code>allowOrigins</code>	Array	Gibt die zulässigen Ursprünge an.
<code>allowCredentials</code>	Boolean	Gibt an, ob Anmeldeinformationen in der CORS-Anforderung enthalten sind.
<code>exposeHeaders</code>	Array	Gibt die Header an, die bereitgestellt werden.
<code>maxAge</code>	Integer	Gibt die Anzahl der Sekunden an, während der der Browser

Eigenschaftename	Typ	Beschreibung
		Preflight-Anfrageergebnisse zwischenspeichern soll.
allowMethods	Array	Gibt die zulässigen HTTP-Methoden an.
allowHeaders	Array	Gibt die zulässigen Header an.

x-amazon-apigateway-cors Beispiel

Das folgende Beispiel zeigt eine CORS-Konfiguration für eine HTTP-API.

```
"x-amazon-apigateway-cors": {
  "allowOrigins": [
    "https://www.example.com"
  ],
  "allowCredentials": true,
  "exposeHeaders": [
    "x-apigateway-header",
    "x-amz-date",
    "content-type"
  ],
  "maxAge": 3600,
  "allowMethods": [
    "GET",
    "OPTIONS",
    "POST"
  ],
  "allowHeaders": [
    "x-apigateway-header",
    "x-amz-date",
    "content-type"
  ]
}
```

x-amazon-apigateway-apiEigenschaft -key-source

Geben Sie die Quelle eines API-Schlüssels an, um die API-Methoden, die einen Schlüssel fordern, zu drosseln. Diese Eigenschaft auf API-Ebene ist vom Typ String. Weitere Informationen zur Methodenkonfigurierung für eine API-Schlüsselabfrage finden Sie unter [the section called “Konfigurieren einer Methode zur Verwendung von API-Schlüsseln mit einer OpenAPI-Definition”](#).

Geben Sie die Quelle des API-Schlüssels für Anforderungen an. Folgende Werte sind zulässig:

- HEADER für den Empfang des API-Schlüssels aus dem X-API-Key-Header einer Anforderung.
- AUTHORIZER für den Empfang des API-Schlüssels von `UsageIdentifierKey` von einem Lambda Genehmiger (ehemals als benutzerdefinierter Genehmiger bezeichnet).

x-amazon-apigateway-apiBeispiel für -key-source

Im folgenden Beispiel wird der X-API-Key-Header als Quelle des API-Schlüssels festgelegt.

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "title" : "Test1"
  },
  "schemes" : [ "https" ],
  "basePath" : "/import",
  "x-amazon-apigateway-api-key-source" : "HEADER",
  .
  .
  .
}
```

OpenAPI 3.0.1

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Test1"
  },
  .
  .
  .
}
```

```
"servers" : [ {
  "url" : "/{basePath}",
  "variables" : {
    "basePath" : {
      "default" : "import"
    }
  }
} ],
"x-amazon-apigateway-api-key-source" : "HEADER",
.
.
.
}
```

x-amazon-apigateway-auth Objekt

Definiert einen Autorisierungstyp, der für die Autorisierung von Methodenaufrufen in API Gateway angewendet werden soll.

Eigenschaftename	Typ	Beschreibung
type	string	Gibt den Autorisierungstyp an. Geben Sie "NONE" für einen offenen Zugriff an. Geben Sie "AWS_IAM" an, um IAM-Berechtigungen zu verwenden. Bei Werten wird die Groß- und Kleinschreibung nicht berücksichtigt.

x-amazon-apigateway-auth Beispiel

Das folgende Beispiel legt den Autorisierungstyp für eine API-Methode fest.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
```

```
"info": {
  "title": "openapi3",
  "version": "1.0"
},
"paths": {
  "/protected-by-iam": {
    "get": {
      "x-amazon-apigateway-auth": {
        "type": "AWS_IAM"
      }
    }
  }
}
```

x-amazon-apigateway-authorizer Objekt

Definiert einen Lambda-Genehmiger, Amazon Cognito-Benutzerpool oder JWT-Genehmiger, der für die Autorisierung von Methodenaufrufen in API Gateway angewendet werden soll. Diese Erweiterung gilt für die Sicherheitsdefinition in [OpenAPI 2](#) und das Sicherheitsschema in [OpenAPI 3](#).

Name der Eigenschaft	Typ	Beschreibung
type	string	Der Typ des Genehmigers. Diese Eigenschaft ist erforderlich. Geben Sie für REST token für einen Autorisierer an APIs, bei dem die Identität des Anrufers in ein Autorisierungstoken eingebettet ist. Geben Sie request für einen Genehmiger an, wobei die Aufruferidentität in den Anforderungsparametern enthalten ist. Geben Sie cognito_user_pools

Name der Eigenschaft	Typ	Beschreibung
		für einen Genehmiger an, der einen Amazon Cognito-Benutzerpool verwendet, um den Zugriff auf Ihre API zu kontrollieren. Geben Sie für HTTP APIs <code>request</code> für einen Lambda-Authenticator mit der Anruferidentität in den Anforderungsparametern an. Geben Sie <code>jwt</code> für einen JWT-Genehmiger an.
<code>authorizerUri</code>	<code>string</code>	Der Uniform Resource Identifier (URI) der Lambda-Funktion des Genehmigers. Die Syntax ist wie folgt: <pre>"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1: <i>account-id</i> :function: <i>auth_function_name</i> /invocations"</pre>

Name der Eigenschaft	Typ	Beschreibung
<code>authorizerCredentials</code>	<code>string</code>	Für den Aufruf des Genehmigers erforderliche Anmeldeinformationen, sofern vorhanden, in Form eines ARN einer IAM-Ausführungsrolle. Zum Beispiel „arn:aws:iam::“. <i>account-id IAM_role</i>
<code>authorizerPayloadFormatVersion</code>	<code>string</code>	Gibt für HTTP das Format der Daten an APIs, die API Gateway an einen Lambda-Autorisierer sendet, und wie API Gateway die Antwort von Lambda interpretiert. Weitere Informationen hierzu finden Sie unter the section called “Nutzlastformatversion” .
<code>enableSimpleResponses</code>	<code>Boolean</code>	Gibt für HTTP an APIs, ob ein request Autorisierer einen booleschen Wert oder eine IAM-Richtlinie zurückgibt. Wird nur für Genehmiger mit <code>authorizerPayloadFormatVersion 2.0</code> unterstützt. Falls aktiviert, gibt die Lambda-Genehmiger-Funktion einen booleschen Wert zurück. Weitere Informationen hierzu finden Sie unter the section called “Lambda-Funktionsantwort für Format 2.0” .

Name der Eigenschaft	Typ	Beschreibung
<code>identitySource</code>	<code>string</code>	Eine CSV-Liste der Mapping-Ausdrücke der Anforderungsparameter als Identitätsquelle. Gilt nur für die Genehmiger des Typs <code>request</code> und <code>jwt</code> .
<code>jwtConfiguration</code>	<code>Object</code>	Gibt den Aussteller und die Zielgruppen für einen JWT-Genehmiger an. Weitere Informationen finden Sie JWTConfiguration in der API-Referenz zu API Gateway Version 2. Wird nur für HTTP unterstützte APIs.
<code>identityValidationExpression</code>	<code>string</code>	Ein regulärer Ausdruck für die Validierung des Tokens als eingehende Identität. Beispiel: <code>"^x-[a-z]+"</code> . Wird nur für TOKEN Autorisierer für REST unterstützt. APIs
<code>authorizerResultTtlInSeconds</code>	<code>string</code>	Die Anzahl der Sekunden, über die hinweg das Genehmiger-Ergebnis zwischengespeichert wird.
<code>providerARNs</code>	Ein Array von <code>string</code>	Eine Liste des Amazon Cognito Cognito-Benutzerpools ARNs für die <code>COGNITO_USER_POOLS</code> .

x-amazon-apigateway-authorizer Beispiele für REST APIs

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen gibt einen Lambda-Genehmiger des Typs "token" des Namens a test-authorizer.

```
"securityDefinitions" : {
  "test-authorizer" : {
    "type" : "apiKey",                                // Required and the value must be
    "apiKey" for an API Gateway API.
    "name" : "Authorization",                          // The name of the header containing
    the authorization token.
    "in" : "header",                                  // Required and the value must be
    "header" for an API Gateway API.
    "x-amazon-apigateway-authtype" : "custom", // Specifies the authorization
    mechanism for the client.
    "x-amazon-apigateway-authorizer" : {              // An API Gateway Lambda authorizer
    definition
      "type" : "token",                               // Required property and the value
      must "token"
      "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
      "authorizerCredentials" : "arn:aws:iam::account-id:role",
      "identityValidationExpression" : "^x-[a-z]+",
      "authorizerResultTtlInSeconds" : 60
    }
  }
}
```

Das folgende OpenAPI-Operations-Objekt-Snippet legt GET /http auf die Verwendung des vorhergehenden Lambda-Genehmigers fest.

```
"/http" : {
  "get" : {
    "responses" : { },
    "security" : [ {
      "test-authorizer" : [ ]
    } ],
    "x-amazon-apigateway-integration" : {
      "type" : "http",
      "responses" : {
```

```

        "default" : {
            "statusCode" : "200"
        }
    },
    "httpMethod" : "GET",
    "uri" : "http://api.example.com"
}
}
}

```

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen spezifiziert einen Lambda-Genehmiger des Typs "request" mit einem einzigen Header-Parameter (auth) als Identitätsquelle. Die securityDefinitions heißt request_authorizer_single_header.

```

"securityDefinitions": {
    "request_authorizer_single_header" : {
        "type" : "apiKey",
        "name" : "auth",                // The name of a single header or query parameter
        // as the identity source.
        "in" : "header",                // The location of the single identity source
        // request parameter. The valid value is "header" or "query"
        "x-amazon-apigateway-authtype" : "custom",
        "x-amazon-apigateway-authorizer" : {
            "type" : "request",
            "identitySource" : "method.request.header.auth", // Request parameter mapping
            // expression of the identity source. In this example, it is the 'auth' header.
            "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
            LambdaRole",
            "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
            functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
            Authorizer:vtwo/invocations",
            "authorizerResultTtlInSeconds" : 300
        }
    }
}

```

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen spezifiziert einen Lambda-Genehmiger des Typs "request" mit einem Header (HeaderAuth1) und einem Abfragezeichenfolgenparameter QueryString1 als Identitätsquellen.

```

"securityDefinitions": {
    "request_authorizer_header_query" : {

```

```

    "type" : "apiKey",
    "name" : "Unused",           // Must be "Unused" for multiple identity sources
    or non header or query type of request parameters.
    "in" : "header",           // Must be "header" for multiple identity sources
    or non header or query type of request parameters.
    "x-amazon-apigateway-authtype" : "custom",
    "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "method.request.header.HeaderAuth1,
method.request.querystring.QueryString1", // Request parameter mapping expressions
of the identity sources.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
    }
}
}

```

Das folgende OpenAPI-Sicherheitsdefinitionsbeispiel gibt einen API Gateway Lambda-Genehmiger vom Typ "request" mit einer einstufigen Variablen (stage) als Identitätsquelle an.

```

"securityDefinitions": {
    "request_authorizer_single_stagevar" : {
        "type" : "apiKey",
        "name" : "Unused",           // Must be "Unused", for multiple identity sources
    or non header or query type of request parameters.
        "in" : "header",           // Must be "header", for multiple identity sources
    or non header or query type of request parameters.
        "x-amazon-apigateway-authtype" : "custom",
        "x-amazon-apigateway-authorizer" : {
            "type" : "request",
            "identitySource" : "stageVariables.stage", // Request parameter mapping
expression of the identity source. In this example, it is the stage variable.
            "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
            "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
            "authorizerResultTtlInSeconds" : 300
        }
    }
}

```

```

    }
  }
}

```

Das folgende Beispiel für eine OpenAPI-Sicherheitsdefinition gibt einen Amazon Cognito-Benutzerpool als Genehmiger an.

```

"securityDefinitions": {
  "cognito-pool": {
    "type": "apiKey",
    "name": "Authorization",
    "in": "header",
    "x-amazon-apigateway-authtype": "cognito_user_pools",
    "x-amazon-apigateway-authorizer": {
      "type": "cognito_user_pools",
      "providerARNs": [
        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-east-1_ABC123"
      ]
    }
  }
}

```

Der folgende OpenAPI-Operationsobjektausschnitt legt GET /http fest, um den vorherigen Amazon Cognito-Benutzerpool als Genehmiger ohne benutzerdefinierte Bereiche zu verwenden.

```

"/http" : {
  "get" : {
    "responses" : { },
    "security" : [ {
      "cognito-pool" : [ ]
    } ],
    "x-amazon-apigateway-integration" : {
      "type" : "http",
      "responses" : {
        "default" : {
          "statusCode" : "200"
        }
      },
      "httpMethod" : "GET",
      "uri" : "http://api.example.com"
    }
  }
}

```

x-amazon-apigateway-authorizer Beispiele für HTTP APIs

Das folgende OpenAPI 3.0-Beispiel erstellt einen JWT-Genehmiger für eine HTTP-API, die Amazon Cognito als Identitätsanbieter verwendet, mit dem Authorization-Header als Identitätsquelle.

```
"securitySchemes": {
  "jwt-authorizer-oauth": {
    "type": "oauth2",
    "x-amazon-apigateway-authorizer": {
      "type": "jwt",
      "jwtConfiguration": {
        "issuer": "https://cognito-idp.region.amazonaws.com/userPoolId",
        "audience": [
          "audience1",
          "audience2"
        ]
      },
      "identitySource": "$request.header.Authorization"
    }
  }
}
```

Im folgenden Beispiel für OpenAPI 3.0 wird derselbe JWT-Genehmiger erstellt wie im vorherigen Beispiel. In diesem Beispiel wird jedoch die OpenAPI-openIdConnectUrl-Eigenschaft verwendet, damit der Aussteller automatisch erkannt wird. Die openIdConnectUrl muss vollständig gebildet sein.

```
"securitySchemes": {
  "jwt-authorizer-autofind": {
    "type": "openIdConnect",
    "openIdConnectUrl": "https://cognito-idp.region.amazonaws.com/userPoolId/.well-known/openid-configuration",
    "x-amazon-apigateway-authorizer": {
      "type": "jwt",
      "jwtConfiguration": {
        "audience": [
          "audience1",
          "audience2"
        ]
      },
      "identitySource": "$request.header.Authorization"
    }
  }
}
```



```
}  
}
```

Das folgende Beispiel erstellt einen Lambda-Genehmiger für eine HTTP-API. Dieser Beispiel-Genehmiger verwendet den Authorization-Header als Identitätsquelle. Der Genehmiger verwendet Nutzlast-Formatversion 2.0 und gibt den booleschen Wert zurück, da `enableSimpleResponses` auf `true` gesetzt ist.

```
"securitySchemes" : {  
  "lambda-authorizer" : {  
    "type" : "apiKey",  
    "name" : "Authorization",  
    "in" : "header",  
    "x-amazon-apigateway-authorizer" : {  
      "type" : "request",  
      "identitySource" : "$request.header.Authorization",  
      "authorizerUri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123456789012:function:function-name/invocations",  
      "authorizerPayloadFormatVersion" : "2.0",  
      "authorizerResultTtlInSeconds" : 300,  
      "enableSimpleResponses" : true  
    }  
  }  
}
```

x-amazon-apigateway-authtype Eigentum

Für REST kann diese Erweiterung verwendet werden APIs, um einen benutzerdefinierten Typ eines Lambda-Autorisierers zu definieren. In diesem Fall liegt der Wert in Freiform vor. Beispielsweise kann eine API mehrere Lambda-Genehmiger haben, die verschiedene interne Schemata verwenden. Sie können diese Erweiterung verwenden, um das interne Schema eines Lambda-Genehmigers zu identifizieren.

In HTTP APIs und REST kann sie häufiger auch verwendet werden APIs, um die IAM-Autorisierung für mehrere Operationen zu definieren, die dasselbe Sicherheitsschema verwenden. In diesem Fall ist der Begriff `awsSigv4` ein reservierter Begriff, zusammen mit jedem Begriff, dem ein Präfix `aws` vorangestellt ist.

Diese Erweiterung gilt für das Sicherheitsschema des Typs `apiKey` in [OpenAPI 2](#) und [OpenAPI 3](#).

x-amazon-apigateway-authtype Beispiel

Das folgende OpenAPI 3-Beispiel definiert die IAM-Genehmigung für mehrere Ressourcen in einer REST-API oder HTTP API:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3",
    "version" : "1.0"
  },
  "paths" : {
    "/operation1" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    },
    "/operation2" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "sigv4Reference" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
```

```

        "x-amazon-apigateway-authtype": "awsSigv4"
    }
}
}
}

```

Das folgende OpenAPI 3-Beispiel definiert einen Lambda-Genehmiger mit einem benutzerdefinierten Schema für eine REST-API:

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3 for REST API",
    "version" : "1.0"
  },
  "paths" : {
    "/protected-by-lambda-authorizer" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "myAuthorizer" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "myAuthorizer" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authorizer" : {
          "identitySource" : "method.request.header.Authorization",
          "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
          "authorizerResultTtlInSeconds" : 300,
          "type" : "request",
          "enableSimpleResponses" : false
        }
      }
    }
  }
}

```

```
    "x-amazon-apigateway-authtype": "Custom scheme with corporate claims"
  }
},
"x-amazon-apigateway-importexport-version" : "1.0"
}
```

Weitere Informationen finden Sie auch unter

[authorizer.authType](#)

x-amazon-apigateway-binaryEigenschaft -media-types

Gibt die Liste der binären Medientypen an, die von API Gateway unterstützt werden sollen, z. B. `application/octet-stream` und `image/jpeg`. Diese Erweiterung ist ein JSON-Array. Dies sollte als Top-Level-Erweiterung des Herstellers im OpenAPI-Dokument eingeschlossen werden.

x-amazon-apigateway-binaryBeispiel für -Medientypen

Das folgende Beispiel zeigt die Codierungs-Suchreihenfolge einer API.

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

x-amazon-apigateway-documentation Objekt

Definiert die Dokumentationssteile, die in API Gateway importiert werden sollen. Dieses Objekt ist ein JSON-Objekt, das ein Array der `DocumentationPart`-Instances enthält.

Eigenschaftename	Typ	Beschreibung
<code>documentationParts</code>	Array	Ein Array der exportierten oder importierten <code>DocumentationPart</code> -Instances.
<code>version</code>	String	Die Versions-ID des Snapshots der exportierten Dokumentationsbausteine.

x-amazon-apigateway-documentation Beispiel

Das folgende Beispiel einer API Gateway-Erweiterung für OpenAPI definiert `DocumentationParts`-Instances, die in eine API in API Gateway importiert oder von einer API in API Gateway exportiert werden sollen.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
      {
        "location": {
          "type": "API"
        },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
        ... // Another DocumentationPart instance
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-Zugriffsmodus

Gibt den Endpunktzugriffsmodus für eine REST-API oder einen benutzerdefinierten Domainnamen an. Weitere Informationen zum Endpunktzugriffsmodus finden Sie unter [Endpunktzugriffsmodus](#).

x-amazon-apigateway-endpoint-access-mode Beispiel für

Im folgenden Beispiel wird der Endpunktzugriffsmodus auf `Strict` festgelegt.

```
"x-amazon-apigateway-endpoint-access-mode": "STRICT"
```

x-amazon-apigateway-endpoint-Konfigurationsobjekt

Gibt Details der Endpunktkonfiguration für eine API an. Diese Erweiterung ist eine erweiterte Eigenschaft des Objekts [OpenAPI-Operation](#). Dieses Objekt sollte in [Top-Level-Anbietererweiterungen](#) für Swagger 2.0 vorhanden sein. Für OpenAPI 3.0 sollte es unter den Anbietererweiterungen des [Server-Objekts](#) vorhanden sein.

Eigenschaftename	Typ	Beschreibung
<code>disableExecuteApiEndpoint</code>	Boolesch	Gibt an, ob Clients Ihre API mithilfe des <code>execute-api</code> - Standardendpunkts aufrufen können. Standardmäßig können Kunden Ihre API mit dem standardmäßigen <code>https://{api_id}.execute-api.{region}.amazonaws.com</code> - Endpunkt aufrufen. Wenn Sie erzwingen möchten, dass Kunden einen benutzerdefinierten Domännennamen verwenden, um Ihre API aufzurufen, geben Sie <code>true</code> an.
<code>vpcEndpointIds</code>	Ein Array von String	Eine Liste von VpcEndpoint Bezeichnern, anhand derer Route 53-Aliaseinträge für eine REST-API erstellt werden sollen. Es wird nur für REST, APIs den PRIVATE Endpunkttyp, unterstützt.
<code>ipAddressType</code>	string	Die IP-Adresstypen, die eine HTTP-API aufrufen können. Wird verwendet <code>ipv4</code> , damit IPv4 Adresstypen eine HTTP-

Eigenschaftename	Typ	Beschreibung
		API aufrufen können. Wird verwendet <code>dualstack</code> , um Typen das Aufrufen einer HTTP-API zu ermöglichen IPv4 und zu IPv6 adressieren. Sie wird nur für HTTP APIs unterstützt.

x-amazon-apigateway-endpoint-Konfigurationsbeispiele

Im folgenden Beispiel werden die angegebenen VPC-Endpunkte der REST-API zugeordnet.

```
"x-amazon-apigateway-endpoint-configuration": {
  "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

Das folgende Beispiel deaktiviert den Standardendpunkt für eine API.

```
"x-amazon-apigateway-endpoint-configuration": {
  "disableExecuteApiEndpoint": true
}
```

Das folgende Beispiel legt den IP-Adresstyp für eine HTTP-API auf Dualstack fest.

```
"x-amazon-apigateway-endpoint-configuration": {
  "ipAddressType": "dualstack"
}
```

x-amazon-apigateway-gatewayObjekt -Responses

Definiert die Gateway-Antworten für eine API als eine String-zu-Zuordnung von [GatewayResponse](#) Schlüssel-Wert-Paaren. Die Erweiterung gilt für die OpenAPI-Struktur auf Stammebene.

Name der Eigenschaft	Typ	Description
<i>responseType</i>	x-amazon-apigateway-gateway-responses.GatewayResponse	A für den angegebenen. GatewayResponse <i>responseType</i>

x-amazon-apigateway-gatewayBeispiel für -Antworten

Die folgende API Gateway Gateway-Erweiterung für das OpenAPI-Beispiel definiert eine [GatewayResponses](#)Map, die zwei [GatewayResponse](#)Instanzen enthält — eine für den DEFAULT_4XX Typ und eine weitere für den Typ. INVALID_API_KEY

```
{
  "x-amazon-apigateway-gateway-responses": {
    "DEFAULT_4XX": {
      "responseParameters": {
        "gatewayresponse.header.Access-Control-Allow-Origin": "'domain.com'"
      },
      "responseTemplates": {
        "application/json": "{\"message\": test 4xx b }"
      }
    },
    "INVALID_API_KEY": {
      "statusCode": "429",
      "responseTemplates": {
        "application/json": "{\"message\": test forbidden }"
      }
    }
  }
}
```

x-amazon-apigateway-gateway-Responses.GatewayResponse-Objekt

Definiert eine Gateway-Antwort eines bestimmten Antworttyps, einschließlich des Statuscodes, sämtlicher geltender Antwortparameter oder Antwortvorlagen.

Eigenschaftename	Typ	Description
<i>responseParameters</i>	x-amazon-apigateway-gateway-Responses.Response-Parameter	Spezifiziert die Parameter , nämlich die GatewayResponseHeader-Parameter . Die Parameterwerte können jeden beliebigen eingehenden request parameter -Wert oder einen statischen benutzerdefinierten Wert übernehmen.
<i>responseTemplates</i>	x-amazon-apigateway-gateway-Responses.ResponseTemplates	Gibt die Mapping-Vorlagen der Gateway-Antwort an. Die Vorlagen werden nicht von der VTL-Engine verarbeitet.
<i>statusCode</i>	string	Ein HTTP-Statuscode für die Gateway-Antwort.

x-amazon-apigateway-gateway-Responses.GatewayResponse-Beispiel

Das folgende Beispiel für die API Gateway Gateway-Erweiterung für OpenAPI definiert, [GatewayResponse](#) dass die INVALID_API_KEY Antwort so angepasst werden soll, dass sie den Statuscode von 456, den api-key Header-Wert der eingehenden Anfrage und eine "Bad api-key" Nachricht zurückgibt.

```
"INVALID_API_KEY": {
  "statusCode": "456",
  "responseParameters": {
    "gatewayresponse.header.api-key": "method.request.header.api-key"
  },
  "responseTemplates": {
    "application/json": "{\"message\": \"Bad api-key\" }"
  }
}
```

x-amazon-apigateway-gateway-Responses.ResponseParameters-Objekt

Definiert eine string-to-string Zuordnung von Schlüssel-Wert-Paaren, um Gateway-Antwortparameter aus den eingehenden Anforderungsparametern oder unter Verwendung von Literalzeichenfolgen zu generieren. Wird nur für REST unterstützt. APIs

Name der Eigenschaft	Typ	Beschreibung
gatewayresponse. <i>param-position</i> <i>.param-name</i>	string	<i>param-position</i> kann header, path oder querystring sein. Weitere Informationen finden Sie unter Parameterzuweisung für REST-APIs in API Gateway .

x-amazon-apigateway-gateway-Responses.ResponseParameters — Beispiel

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt einen Ausdruck für die Zuordnung von [GatewayResponse](#) Antwortparametern, um die CORS-Unterstützung für Ressourcen auf den *.example.domain Domänen zu aktivieren.

```
"responseParameters": {
  "gatewayresponse.header.Access-Control-Allow-Origin": '*.example.domain',
  "gatewayresponse.header.from-request-header" : method.request.header.Accept,
  "gatewayresponse.header.from-request-path" : method.request.path.petId,
  "gatewayresponse.header.from-request-query" : method.request.querystring.qname
}
```

x-amazon-apigateway-gateway-Responses.ResponseTemplates-Objekt

Definiert [GatewayResponse](#) Zuordnungsvorlagen als string-to-string Zuordnung von Schlüssel-Wert-Paaren für eine bestimmte Gateway-Antwort. Für jedes Schlüssel-Wert-Paar ist der Schlüssel der

Inhaltstyp. Zum Beispiel ist "application/json" und der Wert eine "stringify"-Mapping-Vorlage für einfache Variablenersetzungen. Eine GatewayResponse-Mapping-Vorlage wird nicht vom [Velocity Template Language \(VTL\)](#)-Modul verarbeitet.

Eigenschaftename	Typ	Beschreibung
<i>content-type</i>	string	Eine GatewayResponse - Text-Mapping-Vorlage, die nur einfache Variablen ersetzungen unterstützt, um einen Gateway-Antworttext anzupassen.

x-amazon-apigateway-gateway-Responses.ResponseTemplates — Beispiel

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt eine [GatewayResponse](#) Mapping-Vorlage zum Anpassen einer vom API Gateway generierten Fehlerantwort in ein anwendungsspezifisches Format.

```
"responseTemplates": {
  "application/json": "{ \"message\": $context.error.messageString, \"type\": $context.error.responseType, \"statusCode\": '488' }"
```

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt eine [GatewayResponse](#) Mapping-Vorlage zum Überschreiben einer vom API Gateway generierten Fehlerantwort mit einer statischen Fehlermeldung.

```
"responseTemplates": {
  "application/json": "{ \"message\": 'API-specific errors' }"
```

x-amazon-apigateway-importexport-Version

Gibt die Version des API Gateway Gateway-Import- und Exportalgorithmus für HTTP an APIs. Derzeit wird als einziger Wert unterstützt 1.0. Weitere Informationen finden Sie unter [exportVersion](#) in der API Gateway Version 2 API-Referenz.

x-amazon-apigateway-importexportBeispiel für eine Version

Im folgenden Beispiel wird die Import- und Exportversion auf festgelegt 1.0.

```
{
  "openapi": "3.0.1",
  "x-amazon-apigateway-importexport-version": "1.0",
  "info": { ...
```

x-amazon-apigateway-integration Objekt

Gibt die Details der für diese Methode verwendeten Backend-Integration an. Diese Erweiterung ist eine erweiterte Eigenschaft des Objekts [OpenAPI-Operation](#). Das Ergebnis ist ein [API Gateway-Integration](#)-Objekt.

Eigenschaftename	Typ	Beschreibung
cacheKeyParameters	Ein Array von string	Eine Liste an Anforderungsparametern, deren Werte zwischengespeichert werden sollen.
cacheNamespace	string	Eine API-spezifische Tag-Gruppe zugehöriger zwischengespeicherter Parameter.
connectionId	string	Die ID von a VpcLink für die private Integration.
connectionType	string	Der Verbindungstyp der Integration. Der gültige Wert

Eigenschaftename	Typ	Beschreibung
		ist "VPC_LINK" für die private Integration oder andernfalls "INTERNET" .
credentials	string	Geben Sie für rollenbasierte AWS IAM-Anmeldeinformationen den ARN einer entsprechenden IAM-Rolle an. Wenn sie nicht angegeben sind, werden als Anmeldeinformationen standardmäßig ressourcenbasierte Berechtigungen festgelegt, die manuell hinzugefügt werden, damit die API auf die Ressource zugreifen kann. Weitere Informationen finden Sie unter Gewähren von Berechtigungen mit einer Ressourcennrichtlinie. Hinweis: Stellen Sie bei der Verwendung von IAM-Anmeldeinformationen sicher, dass regionale AWS -STS-Endpunkte für die Region aktiviert sind, in der diese API für maximale Leistung bereitgestellt wird.

Eigenschaftename	Typ	Beschreibung
contentHandling	string	Fordern Sie Nutzlast-Kodierungs-Umwandlungstypen an. Gültige Werte sind 1) <code>CONVERT_TO_TEXT</code> für die Umwandlung einer binären Nutzlast in eine base64-kodierte Zeichenfolge oder die Umwandlung einer Textnutzlast in eine utf-8-kodierte Zeichenfolge oder die native Weiterleitung der Textnutzlast ohne Änderung und 2) <code>CONVERT_TO_BINARY</code> für die Umwandlung einer Textnutzlast in einen base64-dekodierten Blob oder die native Weiterleitung einer binären Nutzlast ohne Änderung.
httpMethod	string	Die in der Integrationsanforderung verwendete HTTP-Methode. Für Lambda-Funktionsaufrufe muss der Wert <code>POST</code> sein.
integrationSubtype	string	Gibt den Integrationsuntertyp für eine Serviceintegration an. AWS wird nur für HTTP APIs unterstützt. Unterstützte Integration-Subtypen finden Sie unter the section called “AWSReferenz für -Serviceintegrationen” .

Eigenschaftename	Typ	Beschreibung
integrationTarget	string	Der ALB- oder NLB-Listener, an den die Anfrage gesendet werden soll. Wird nur für private Integrationen unterstützt, die VPC-Links V2 verwenden. Weitere Informationen finden Sie unter the section called “VPC-Verbindungen V2” .
passthroughBehavior	string	Gibt an, wie eine Anforderungsnutzlast eines nicht zugeordneten Inhaltstyps unverändert an die Integrationsanforderung weitergeleitet wurde. Unterstützte Werte sind <code>when_no_templates</code> , <code>when_no_match</code> und <code>never</code> . Weitere Informationen finden Sie unter Integration.passthroughBehavior .

Eigenschaftename	Typ	Beschreibung
payloadFormatVersion	string	Gibt das Format der an eine Integration gesendeten Nutzlast an. Für HTTP erforderlich. APIs für HTTP sind APIs, die unterstützten Werte für Lambda-Proxyintegrationen 1.0 und 2.0. Für alle anderen Integrationen ist 1.0 der einzige unterstützte Wert. Weitere Informationen hierzu finden Sie unter the section called “AWS Lambda Integrationen” und the section called “AWSReferenz für -Serviceintegrationen” .

Eigenschaftename	Typ	Beschreibung
<code>requestParameters</code>	x-amazon-apigateway-integration.RequestParameters Objekt	Gibt für REST APIs Zuordnungen von Methodenanforderungsparametern zu Integrationsanforderungsparametern an. Die unterstützten Anforderungsparameter sind <code>queryString</code> , <code>path</code> , <code>header</code> und <code>body</code>. Für HTTP sind Anforderungsparameter eine Schlüssel-Wert-Zuordnung APIs, die Parameter spezifiziert, die an <code>AWS_PROXY</code> Integrationen mit einem bestimmten Wert übergeben werden. <code>integrationSubtype</code> Sie können statische Werte angeben oder Anforderungsdaten, Stufenvariablen oder Kontextvariablen zuordnen, die zur Laufzeit ausgewertet werden. Weitere Informationen hierzu finden Sie unter the section called “AWS Serviceintegrationen”.
<code>requestTemplates</code>	x-amazon-apigateway-integration.requestTemplates -Objekt	Mapping-Vorlagen für eine Anforderungsnutzlast angegebener MIME-Typen.

Eigenschaftename	Typ	Beschreibung
<code>responses</code>	x-amazon-apigateway-integration.responses-Objekt	Definiert die Antworten der Methode und gibt gewünschte Parameter-Mappings oder Nutzlast-Mappings von Integrationsantworten auf Methodenantworten an.
<code>responseTransferMode</code>	<code>string</code>	Der Antwortübertragungsmodus der Integration. Wird verwendet <code>BUFFERED</code> , damit API Gateway auf den Empfang der vollständigen Antwort wartet, bevor es mit der Übertragung beginnt. Wird verwendet <code>STREAM</code> , damit API Gateway Teilantworten an den Client zurücksendet, sobald sie verfügbar sind. Weitere Informationen hierzu finden Sie unter the section called “Streamen der Integrationsantwort für Ihre Proxy-Integrationen” .
<code>timeoutInMillis</code>	<code>integer</code>	Integrationszeitüberschreitungen zwischen 50 ms und 29.000 ms.

Eigenschaftename	Typ	Beschreibung
type	string	Der Typ der Integration mit dem angegebenen Backend. Folgende Werte sind zulässig: • <code>http</code> oder <code>http_proxy</code> , für die Integration mit einem HTTP-Backend. • <code>aws_proxy</code> , für die Integration mit AWS Lambda-Funktionen. • <code>aws</code>, für die Integration mit AWS Lambda-Funktionen oder anderen AWS Diensten wie Amazon DynamoDB, Amazon Simple Notification Service oder Amazon Simple Queue Service. • <code>mock</code> für die Integration mit API Gateway, ohne ein Backend aufzurufen. Weitere Informationen über die Integrationstypen finden Sie unter integration:type.
tlsConfig	the section called “x-amazon-apigateway-integration.tlsConfig”	Gibt die TLS-Konfiguration für eine Integration an.

Eigenschaftename	Typ	Beschreibung
uri	string	Die Endpunkt-URI des Backends. Für Integrationen des aws-Typs ist es ein ARN-Wert. Für die HTTP-Integration ist dies die URL des HTTP-Endpunkts einschließlich des https- oder http-Schemas.

x-amazon-apigateway-integration Beispiele

Für HTTP APIs können Sie Integrationen im Komponentenbereich Ihrer OpenAPI-Definition definieren. Weitere Informationen hierzu finden Sie unter [x-amazon-apigateway-integrations Objekt](#).

```
"x-amazon-apigateway-integration": {  
  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"  
}
```

Im folgenden Beispiel wird eine Integration mit einer Lambda-Funktion erstellt. Zu Demonstrationszwecken wird für die Beispiel-Mapping-Vorlagen, die in requestTemplates und responseTemplates der Beispiele unten gezeigt werden, angenommen, dass die folgenden JSON-formatierten Nutzlasten angewendet werden: { "name": "value_1", "key": "value_2", "redirect": { "url" : "..."} } zur Generierung einer JSON-Ausgabe von { "stage": "value_1", "user-id": "value_2" } oder einer XML-Ausgabe von <stage>value_1</stage>.

```
"x-amazon-apigateway-integration" : {  
  "type" : "aws",  
  "uri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-east-1:012345678901:function:HelloWorld/invocations",  
  "httpMethod" : "POST",  
  "credentials" : "arn:aws:iam::012345678901:role/apigateway-invoke-lambda-exec-  
role",  
  "requestTemplates" : {
```

```

        "application/json" : "#set ($root=$input.path('$')) { \"stage\":
\"$root.name\", \"user-id\": \"$root.key\" }",
        "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
    },
    "requestParameters" : {
        "integration.request.path.stage" : "method.request.querystring.version",
        "integration.request.querystring.provider" :
"method.request.querystring.vendor"
    },
    "cacheNamespace" : "cache namespace",
    "cacheKeyParameters" : [],
    "responses" : {
        "2\\d{2}" : {
            "statusCode" : "200",
            "responseParameters" : {
                "method.response.header.requestId" : "integration.response.header.cid"
            },
            "responseTemplates" : {
                "application/json" : "#set ($root=$input.path('$')) { \"stage\":
\"$root.name\", \"user-id\": \"$root.key\" }",
                "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
            }
        },
        "302" : {
            "statusCode" : "302",
            "responseParameters" : {
                "method.response.header.Location" :
"integration.response.body.redirect.url"
            }
        },
        "default" : {
            "statusCode" : "400",
            "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static value'"
            }
        }
    }
}

```

Beachten Sie, dass doppelte Anführungszeichen (") für die JSON-Zeichenfolge in den Mapping-Vorlagen durch einen umgekehrten Schrägstrich (\") geschützt sein müssen.

x-amazon-apigateway-integrations Objekt

Definiert eine Sammlung von Integrationen. Sie können Integrationen im Komponentenbereich Ihrer OpenAPI-Definition definieren und die Integrationen für mehrere Routen wiederverwenden. Wird nur für HTTP unterstützte APIs.

Name der Eigenschaft	Typ	Description
<i>integration</i>	x-amazon-apigateway-integration Objekt	Eine Sammlung von Integrationsobjekten.

x-amazon-apigateway-integrations Beispiel

Im folgenden Beispiel wird eine HTTP-API erstellt, die zwei Integrationen definiert und die Integrationen über referenziert `$ref`: `"#/components/x-amazon-apigateway-integrations/integration-name".`

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Integrations",
    "description": "An API that reuses integrations",
    "version": "1.0"
  },
  "servers": [
    {
      "url": "https://example.com/{basePath}",
      "description": "The production API server",
      "variables": {
        "basePath": {
          "default": "example/path"
        }
      }
    }
  ],
}
```

```

"paths":
{
  "/":
  {
    "get":
    {
      "x-amazon-apigateway-integration":
      {
        "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
      }
    }
  },
  "/pets":
  {
    "get":
    {
      "x-amazon-apigateway-integration":
      {
        "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
      }
    }
  },
  "/checkout":
  {
    "get":
    {
      "x-amazon-apigateway-integration":
      {
        "$ref": "#/components/x-amazon-apigateway-integrations/integration2"
      }
    }
  }
},
"components": {
  "x-amazon-apigateway-integrations":
  {
    "integration1":
    {
      "type": "aws_proxy",
      "httpMethod": "POST",
      "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",

```

```
        "passthroughBehavior": "when_no_templates",
        "payloadFormatVersion": "1.0"
    },
    "integration2":
    {
        "type": "aws_proxy",
        "httpMethod": "POST",
        "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:example-function/invocations",
        "passthroughBehavior": "when_no_templates",
        "payloadFormatVersion" : "1.0"
    }
}
}
```

x-amazon-apigateway-integration.requestTemplates-Objekt

Gibt die Mapping-Vorlagen für eine Anforderungsnutzlast der angegebenen MIME-Typen an.

Eigenschaftename	Typ	Beschreibung
<i>MIME type</i>	string	Ein Beispiel für den MIME-Typ ist application/json. Weitere Informationen zum Erstellen einer Mapping-Vorlage finden Sie unter the section called “Transformationen mit Zuweisungsvorlagen” .

x-amazon-apigateway-integration.requestTemplates Beispiel

Im folgenden Beispiel werden Mapping-Vorlagen für eine Anforderungsnutzlast der application/json- und application/xml-MIME-Typen festgelegt.


```
"requestTemplates" : {
  "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
\"user-id\": \"$root.key\" }",
  "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration.RequestParameters Objekt

Gibt für REST APIs Zuordnungen von benannten Methodenanforderungsparametern zu Integrationsanforderungsparametern an. Die Methodenanforderungsparameter müssen definiert werden, bevor auf sie verwiesen wird.

Gibt für HTTP Parameter an APIs, die an AWS_PROXY Integrationen mit einem bestimmten Wert übergeben werden. `integrationSubtype`

Name der Eigenschaft	Typ	Description
<code>integration.requestParameters.<param-type>.<param-name></code>	string	Für REST APIs ist der Wert in der Regel ein vordefinierter Methodenanforderungsparameter des <code>method.requestParameters.<param-type>.<param-name></code> Formats, wobei <code>queryString</code> , <code>pathheader</code> , oder <code>body</code> sein <code><param-type></code> kann. <code>\$context.VARIABLE_NAME</code> , <code>\$stageVariables.VARIABLE_NAME</code> und <code>STATIC_VALUE</code> sind jedoch ebenfalls gültig. Für den body-Parameter ist der <code><param-name></code> ein JSON-Pfadausdruck ohne das <code>\$.-</code> Präfix.

Name der Eigenschaft	Typ	Description
<i>parameter</i>	string	Für HTTP sind Anforderungsparameter eine Key-Value-Map APIs, die Parameter spezifiziert, die an AWS_PROXY Integrationen mit einem bestimmten Wert übergeben werden. <code>integrationSubtype</code> Sie können statische Werte angeben oder Anforderungsdaten, Stufenvariablen oder Kontextvariablen zuordnen, die zur Laufzeit ausgewertet werden. Weitere Informationen hierzu finden Sie unter the section called "AWS Serviceintegrationen" .

x-amazon-apigateway-integration.requestParameters Beispiel für

Im folgenden Beispiel für Anforderungsparameter-Mappings werden die Abfrage- (`version`), Header- (`x-user-id`) und Pfad- (`service`) Parameter einer Methodenanforderung in die Abfrage- (`stage`), Header- (`x-userid`) und Pfad- (`op`) Parameter der Integrationsanforderung übersetzt.

Note

Wenn Sie Ressourcen über OpenAPI oder erstellen CloudFormation, sollten statische Werte in einfache Anführungszeichen gesetzt werden.

Um diesen Wert über die Konsole hinzuzufügen, geben Sie `application/json` ohne Anführungszeichen in das Feld ein.

```
"requestParameters" : {
```

```
"integration.request.querystring.stage" : "method.request.querystring.version",
"integration.request.header.x-userid" : "method.request.header.x-user-id",
"integration.request.path.op" : "method.request.path.service"
},
```

x-amazon-apigateway-integration.responses-Objekt

Definiert die Antworten der Methode und gibt Parameter-Mappings oder Nutzlast-Mappings von Integrationsantworten auf Methodenantworten an.

Eigenschaftensname	Typ	Description
<i>Response status pattern</i>	x-amazon-apigateway-integration.response-Objekt	Entweder ein regulärer Ausdruck, der verwendet wird, um die Integrationsantwort mit der Methodenantwort abzugleichen, oder default, um eine Antwort zu erfassen, die noch nicht konfiguriert wurde. Für HTTP-Integrationen gilt der Regex für den Integrationsantwort-Statuscode. Bei Lambda-Aufrufen gilt der reguläre Ausdruck für das errorMessage-Feld des Fehlerinformationsobjekts, das AWS Lambda als Fehlerantworttext zurückgegeben wird, wenn die Ausführung der Lambda-Funktion eine Ausnahme auslöst.


```
    }
  },
  "302" : {
    "statusCode" : "302",
    "responseParameters" : {
      "method.response.header.Location": "integration.response.body.redirect.url"
    }
  }
}
```

x-amazon-apigateway-integration.response-Objekt

Definiert eine Antwort und gibt Parameter-Mappings oder Nutzlast-Mappings aus der Integrationsantwort in die Methodenantwort an.

Eigenschaftename	Typ	Beschreibung
statusCode	string	HTTP-Statuscode für die Methodenantwort, zum Beispiel, "200". Dieser muss einer übereinstimmenden Antwort im OpenAPI-Operation responses -Feld entsprechen.
responseTemplates	x-amazon-apigateway-integration.responseTemplates-Objekt	Gibt MIME-typenspezifische Mapping-Vorlagen für die Nutzlast der Antwort an.
responseParameters	x-amazon-apigateway-integration.responseParameters-Objekt	Gibt Parameter-Mappings für die Antwort an. Nur die header- und body-Parameter der Integrationsantwort können den header-Parametern der Methode zugeordnet werden.

Eigenschaftename	Typ	Beschreibung
contentHandling	string	Umwandlungstypen für Antwortnutzlastenkodierung. Gültige Werte sind 1) <code>CONVERT_TO_TEXT</code> für die Umwandlung einer binären Nutzlast in eine base64-kodierte Zeichenfolge oder die Umwandlung einer Textnutzlast in eine utf-8-kodierte Zeichenfolge oder die native Weiterleitung der Textnutzlast ohne Änderung und 2) <code>CONVERT_TO_BINARY</code> für die Umwandlung einer Textnutzlast in einen base64-dekodierten Blob oder die native Weiterleitung einer binären Nutzlast ohne Änderung.

x-amazon-apigateway-integration.responseBeispiel für

Im folgenden Beispiel wird eine 302-Antwort für die Methode definiert, die eine Nutzlast des `application/json`- oder `application/xml`-MIME-Typs vom Backend ableitet. Die Antwort verwendet die bereitgestellten Mapping-Vorlagen und gibt die Umleitungs-URL von der Integrationsantwort im `Location`-Header der Methode zurück.

```
{
  "statusCode" : "302",
  "responseTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\", \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
  },
}
```

```
"responseParameters" : {  
    "method.response.header.Location": "integration.response.body.redirect.url"  
}  
}
```

x-amazon-apigateway-integration.responseTemplates-Objekt

Gibt die Mapping-Vorlagen für eine Antwortnutzlast der angegebenen MIME-Typen an.

Eigenschaftename	Typ	Beschreibung
<i>MIME type</i>	string	Gibt eine Mapping-Vorlage an, um den Integrationsantworttext für einen bestimmten MIME-Typ in den Methodenantworttext umzuwandeln. Weitere Informationen zum Erstellen einer Mapping-Vorlage finden Sie unter the section called “Transformationen mit Zuweisungsvorlagen” . Ein Beispiel für das ist. <i>MIME type</i> application/json

x-amazon-apigateway-integration.responseTemplate-Beispiel

Im folgenden Beispiel werden Mapping-Vorlagen für eine Anforderungsnutzlast der application/json- und application/xml-MIME-Typen festgelegt.

```
"responseTemplates" : {  
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",  
    \"user-id\": \"$root.key\" }",
```

```
"application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
```

```
}
```

x-amazon-apigateway-integration.responseParameters-Objekt

Gibt Mappings von Integrationsmethoden-Antwortparameter auf Methodenantwortparameter an. Sie können header, body oder statische Werte dem header-Typ der Methodenantwort zuordnen. Wird nur für REST unterstützt. APIs

Name der Eigenschaft	Typ	Description
method.response.header. <i><param-name></i>	string	Der benannte Parameterwert kann von den header- und body-Typen der Integrationsantwortparameter abgeleitet werden.

x-amazon-apigateway-integration.responseParametersBeispiel für

Im folgenden Beispiel werden der body- und header-Parameter der Integrationsantwort auf zwei header-Parameter der Methodenantwort zugewiesen.

```
"responseParameters" : {  
  "method.response.header.Location" : "integration.response.body.redirect.url",  
  "method.response.header.x-user-id" : "integration.response.header.x-userid"  
}
```

x-amazon-apigateway-integration.tlsConfig-Objekt

Gibt die TLS-Konfiguration für eine Integration an.

Eigenschaftename	Typ	Description
<code>insecureSkipVerification</code>	Boolean	Wird nur für REST unterstützt. APIs Gibt an, ob API Gateway die Überprüfung der Ausstellung des Zertifikats für einen Integrationsendpunkt durch eine unterstützte Zertifizierungsstelle überspringt oder nicht. Dies wird nicht empfohlen, ermöglicht Ihnen aber, Zertifikate zu verwenden, die von privaten Zertifizierungsstellen signiert sind, oder Zertifikate, die selbstsigniert sind. Wenn diese Option aktiviert ist, führt API Gateway dennoch eine grundlegende Zertifikatüberprüfung durch, wobei Ablaufdatum, Hostname und das Vorhandensein einer Stammzertifizierungsstelle des Zertifikats geprüft werden. Das Stammzertifikat der privaten Behörde muss die folgenden Einschränkungen erfüllen: • Die x509-Erweiterung <code>keyUsage</code> muss <code>keyCertSign</code> haben. • Die x509-Erweiterung <code>basicConstraints</code> muss <code>CA:TRUE</code> haben.

Eigenschaftename	Typ	Description
		Wird nur für HTTP- und HTTP_PROXY -Integrationen unterstützt.  Warning Aktivieren von <code>insecureSkipVerification</code> wird nicht empfohlen, insbesondere für Integrationen mit öffentlichen HTTPS-Endpunkten. Wenn Sie diese Option aktivieren <code>insecureSkipVerification</code>, erhöhen Sie das Risiko von man-in-the-middle Angriffen.
<code>serverNameToVerify</code>	<code>string</code>	Wird nur für private HTTP-API-Integrationen unterstützt. Wenn Sie einen Servernamen angeben, verwendet API Gateway diesen, um den Hostnamen auf dem Integrationszertifikat zu überprüfen. Der Servername ist auch im TLS-Handshake enthalten, um Server Name Indication (SNI, Servernamenanzeige) oder virtuelles Hosting zu unterstützen.

x-amazon-apigateway-integration.tlsConfig-Beispiele

Das folgende OpenAPI 3.0-Beispiel aktiviert `insecureSkipVerification` für eine REST-API HTTP-Proxy-Integration.

```
"x-amazon-apigateway-integration": {
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
  "responses": {
    "default": {
      "statusCode": "200"
    }
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "ANY",
  "tlsConfig" : {
    "insecureSkipVerification" : true
  }
  "type": "http_proxy",
}
```

Das folgende OpenAPI 3.0-Beispiel gibt einen `serverNameToVerify` für eine private HTTP-API-Integration an.

```
"x-amazon-apigateway-integration" : {
  "payloadFormatVersion" : "1.0",
  "connectionId" : "abc123",
  "type" : "http_proxy",
  "httpMethod" : "ANY",
  "uri" : "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65",
  "connectionType" : "VPC_LINK",
  "tlsConfig" : {
    "serverNameToVerify" : "example.com"
  }
}
```

x-amazon-apigateway-minimum-Kompressionsgröße

Gibt die minimale Komprimierungsgröße für eine REST-API an. Um die Komprimierung zu aktivieren, geben Sie eine ganze Zahl zwischen 0 und 10485760 an. Weitere Informationen hierzu finden Sie unter [Payload-Komprimierung für REST APIs in API Gateway](#).

x-amazon-apigateway-minimum Beispiel für eine Kompressionsgröße

Das folgende Beispiel gibt eine minimale Komprimierungsgröße von 5242880 Bytes für eine REST-API an.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Gibt eine Ressourcen-Richtlinie für eine REST-API an. Weitere Informationen zu den Ressourcenrichtlinien finden Sie unter [Zugriff auf eine REST-API mit API Gateway-Ressourcenrichtlinien steuern](#). Beispiele für Ressourcenrichtlinien finden Sie unter [Beispiele für API Gateway-Ressourcenrichtlinien](#).

x-amazon-apigateway-policy Beispiel für

Das folgende Beispiel gibt eine Ressourcen-Richtlinie für eine REST-API an. Die Ressourcenrichtlinie verweigert (blockiert) eingehenden Datenverkehr von einem angegebenen Quell-IP-Adressblock an eine API. Wird beim Import unter Verwendung der aktuellen Region in Ihre AWS Konto-ID und die aktuelle REST-API-ID konvertiert. "execute-api:/*" arn:aws:execute-api:*region*:*account-id*:*api-id*/*

```
"x-amazon-apigateway-policy": {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": [
        "execute-api:/*"
      ]
    }
  ]
}
```

```

        "Condition" : {
            "IpAddress": {
                "aws:SourceIp": "192.0.2.0/24"
            }
        }
    }
}
]
}

```

x-amazon-apigateway-request-validator-Eigenschaft

Gibt eine Anforderungsvalidierung an, indem auf einen *request_validator_name* der [x-amazon-apigateway-request-validators-Objekt](#)-Zuweisung verwiesen wird, um die Anforderungsvalidierung auf der enthaltenen API oder einer Methode zu aktivieren. Der Wert dieser Erweiterung ist eine JSON-Zeichenfolge.

Diese Erweiterung kann auf API-Ebene oder auf Methodenebene angegeben werden. Die Validierung auf API-Ebene gilt für alle Methoden, es sei denn, sie wird von der Validierung auf Methodenebene außer Kraft gesetzt.

x-amazon-apigateway-request-validator Beispiel für

Im folgenden Beispiel wird die basic-Anforderungsvalidierung auf API-Ebene angewendet, während die parameter-only-Anforderungsvalidierung auf der POST /validation-Anforderung angewendet wird.

OpenAPI 2.0

```

{
  "swagger": "2.0",
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "basic",

```

```
"paths": {
  "/validation": {
    "post": {
      "x-amazon-apigateway-request-validator" : "params-only",
      ...
    }
  }
}
```

x-amazon-apigateway-request-validators-Objekt

Definiert die unterstützten Anforderungsvalidierungen für die enthaltene API als eine Zuweisung zwischen einem Validierungsnamen und den zugehörigen Anforderungsvalidierungsregeln. Diese Erweiterung gilt für eine REST-API.

Eigenschaftename	Typ	Beschreibung
<i>request_validator_name</i>	x-amazon-apigateway-request-Validators.RequestValidator-Objekt	Gibt die Validierungsregeln an, die aus der benannten Validierung bestehen. Zum Beispiel: <pre>"basic" : { "validate RequestBody" : true, "validate RequestParameters" : true },</pre> Um diese Validierung auf eine bestimmte Methode anzuwenden, muss der Validierungsname (basic) als der Wert der x-amazon-apigateway-request-validator-Eigenschaft-Eigenschaft referenziert werden.

x-amazon-apigateway-request-validatorsBeispiel für

Das folgende Beispiel zeigt eine Reihe von Anforderungsvalidierungen für eine API als eine Zuweisung zwischen einem Validierungsnamen und den zugehörigen Anforderungsvalidierungsregeln.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  ...
}
```

x-amazon-apigateway-request-Validators.RequestValidator-Objekt

Gibt die Validierungsregeln einer Anforderungsvalidierung als Teil der [x-amazon-apigateway-request-validators-Objekt](#)-Zuweisungsdefinition an.

Eigenschaftename	Typ	Beschreibung
validateRequestBody	Boolean	Gibt an, ob der Anforderungstext (<code>true</code>) oder nicht (<code>false</code>) validiert werden soll.
validateRequestParameters	Boolean	Gibt an, ob die erforderlichen Anforderungsparameter (<code>true</code>) oder nicht (<code>false</code>) validiert werden.

x-amazon-apigateway-request-validators.requestValidator

Beispiel für

Das folgende Beispiel zeigt eine Anforderungsvalidierung nur für den Parameter:

```
"params-only": {  
  "validateRequestBody" : false,  
  "validateRequestParameters" : true  
}
```

x-amazon-apigateway-security-Richtlinie

Gibt eine Sicherheitsrichtlinie für eine REST-API an. Wenn Sie eine Sicherheitsrichtlinie erstellen, die mit `SecurityPolicy_` beginnt, müssen Sie auch den [Endpunktzugriffsmodus](#) festlegen. Weitere Informationen zu Sicherheitsrichtlinien finden Sie unter [Sicherheitsrichtlinien für REST-APIs in API Gateway](#).

x-amazon-apigateway-security-policy

Beispiel für

Das folgende Beispiel spezifiziert `SecurityPolicy_TLS13_1_3_2025_0` für eine REST-API.

```
"x-amazon-apigateway-security-policy": "SecurityPolicy_TLS13_1_3_2025_09"
```

x-amazon-apigateway-tagEigenschaft -value

Gibt den Wert eines [AWS -Tags](#) für eine HTTP-API an. Sie können die `x-amazon-apigateway-tag-value` Eigenschaft als Teil des [OpenAPI-Tag-Objekts auf Stammebene verwenden, um AWS Tags für eine HTTP-API](#) anzugeben. Wenn Sie einen Tag-Namen ohne die `x-amazon-apigateway-tag-value`-Eigenschaft angeben, erstellt API Gateway für einen Wert ein Tag mit einer leeren Zeichenfolge.

Weitere Informationen zum Taggen finden Sie unter [API Gateway-Ressourcen taggen](#).

Eigenschaftensname	Typ	Beschreibung
name	String	Gibt den Tag-Schlüssel an.

Eigenschaftename	Typ	Beschreibung
x-amazon-apigateway-tag-value	String	Gibt den Tag-Wert an.

x-amazon-apigateway-tag-value Beispiel für

Das folgende Beispiel gibt zwei Tags für eine HTTP-API an:

- "Owner": "Admin"
- "Prod": ""

```
"tags": [  
  {  
    "name": "Owner",  
    "x-amazon-apigateway-tag-value": "Admin"  
  },  
  {  
    "name": "Prod"  
  }  
]
```

Sicherheit in Amazon API Gateway

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das [Modell der übergreifenden Verantwortlichkeit](#) beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#). Weitere Informationen zu den Compliance-Programmen, die für Amazon API Gateway gelten, finden Sie unter [AWS Services im Umfang nach Compliance-Programm AWS](#).
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation erläutert, wie das Modell der geteilten Verantwortung bei der Verwendung von API Gateway angewendet werden kann. Die folgenden Themen zeigen Ihnen, wie Sie API Gateway konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS Dienste verwenden können, die Ihnen helfen, Ihre API-Gateway-Ressourcen zu überwachen und zu sichern.

Weitere Informationen finden Sie unter [Sicherheit in Amazon API Gateway](#).

Topics

- [Datenschutz in Amazon API Gateway](#)
- [Identitäts- und Zugriffsverwaltung für Amazon API Gateway](#)
- [Protokollierung und Überwachung in Amazon API Gateway](#)
- [Compliance-Validierung für Amazon API Gateway](#)
- [Zuverlässigkeit in Amazon API Gateway](#)
- [Infrastruktursicherheit in Amazon API Gateway](#)
- [Schwachstellenanalyse in Amazon API Gateway](#)

- [Bewährte Methoden in Amazon API Gateway](#)

Datenschutz in Amazon API Gateway

Das AWS [Modell](#) der gilt für den Datenschutz in Amazon API Gateway. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#) . Weitere Informationen zum Datenschutz in Europa finden Sie im [Zentrum für die Datenschutz-Grundverordnung \(DSGVO\)](#).

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit

API Gateway oder anderen Geräten arbeiten und die Konsole, die API oder AWS SDKs AWS-Services verwenden. AWS CLI Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung in Amazon API Gateway

Datenschutz bezieht sich auf den Schutz von Daten während der Übertragung (auf dem Weg zu und von API Gateway) und im Ruhezustand (während sie in der gespeichert werde AWS).

Datenverschlüsselung im Ruhezustand in Amazon API Gateway

Wenn Sie das Caching für eine REST-API aktivieren, können Sie die Cache-Verschlüsselung aktivieren. Weitere Informationen hierzu finden Sie unter [Cache-Einstellungen für REST APIs in API Gateway](#).

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag [AWS Shared Responsibility Model and GDPR](#) im AWS-Sicherheitsblog.

Verschlüsselung und Entschlüsselung Ihres privaten Zertifikatschlüssels

Wenn Sie einen benutzerdefinierten Domainnamen für private APIs erstellen, werden Ihr ACM-Zertifikat und Ihr privater Schlüssel mithilfe eines AWS-verwalteten KMS-Schlüssels mit dem Alias `aws/acm` verschlüsselt. Sie können die Schlüssel-ID mit diesem Alias in der AWS KMS-Konsole unter `AWS-verwaltete Schlüssel` einsehen.

API Gateway greift nicht direkt auf Ihre ACM-Ressourcen zu. Es verwendet den TLS Connection Manager von AWS, um die privaten Schlüssel für Ihr Zertifikat zu sichern und auf sie zuzugreifen. Wenn Sie Ihr ACM-Zertifikat verwenden, um einen benutzerdefinierten API-Gateway-Domainnamen für private APIs zu erstellen, ordnet API Gateway Ihr Zertifikat dem TLS Connection Manager von AWS zu. Dazu erstellen Sie eine Erteilung unter AWS KMS gegenüber Ihrem AWS-verwalteten Schlüssel mit dem Präfix `aws/acm`. Eine Erteilung ist ein Richtlinieninstrument, das es TLS Connection Manager erlaubt, KMS-Schlüssel in kryptografischen Operationen zu verwenden. Die Erteilung erlaubt es dem Empfänger-Prinzipal (TLS Connection Manager), die angegebenen Genehmigungsoperationen für den KMS-Schlüssel aufzurufen, um den privaten Schlüssel Ihres Zertifikats zu entschlüsseln. TLS Connection Manager verwendet dann das Zertifikat und den entschlüsselten privaten (Klartext) Schlüssel für den Aufbau einer sicheren Verbindung (SSL/TLS-

Sitzung) mit Kunden von API-Gateway-Diensten. Die Erteilung wird außer Betrieb genommen, wenn das Zertifikat von einem benutzerdefinierten API-Gateway-Domainnamen für private APIs getrennt wird.

Wenn Sie den Zugriff auf den KMS-Schlüssel deaktivieren möchten, sollten Sie das Zertifikat mithilfe des AWS-Managementkonsole- oder `update-service`-Befehls im Dienst zu ersetzen oder zu entfernen.

Verschlüsselungskontext für API Gateway

Ein [Verschlüsselungskontext](#) ist ein optionaler Satz von Schlüssel-Wert-Paaren, die Kontextinformationen zur möglichen Verwendung Ihres privaten Schlüssels enthalten. AWS KMS bindet den Verschlüsselungskontext an die verschlüsselten Daten und verwendet diese als zusätzliche Authentifizierungsdaten zur Unterstützung der authentifizierten Verschlüsselung.

Wenn Ihre TLS-Schlüssel mit API Gateway und TLS Connection Manager verwendet werden, wird der Name Ihres API-Gateway-Dienstes im Verschlüsselungskontext aufgenommen, der zur Verschlüsselung Ihres Schlüssels im Ruhezustand verwendet wird. Sie können überprüfen, für welchen benutzerdefinierten API-Gateway-Domainnamen Ihr Zertifikat und Ihr privater Schlüssel verwendet werden, indem Sie den Verschlüsselungskontext in Ihren CloudTrail-Protokollen wie im nächsten Abschnitt gezeigt einsehen, oder indem Sie in der ACM-Konsole die Registerkarte Zugeordnete Ressourcen einsehen.

Zur Entschlüsselung von Daten wird derselbe Verschlüsselungskontext in der Anforderung übergeben. API Gateway verwendet denselben Verschlüsselungskontext in allen kryptografischen AWS-KMS-Operationen, wobei der Schlüssel `aws:apigateway:arn` ist und der Wert des Amazon-Ressourcennamens (ARN) der API-Gateway-Ressource `PrivateDomainName` ist.

Im folgenden Beispiel sehen Sie den Verschlüsselungskontext in der Ausgabe einer Operation wie `CreateGrant`.

```
"constraints": {
  "encryptionContextEquals": {
    "aws:acm:arn": "arn:aws:acm:us-west-2:859412291086:certificate/9177097a-f0ae-4be1-93b1-19f911ea4f88",
    "aws:apigateway:arn": "arn:aws:apigateway:us-west-2:859412291086:/domainnames/denyttest-part1.pdx.sahig.people.aws.dev+cbaeumzjhg"
  },
  "operations": [
```

```
"Decrypt"  
],  
"granteePrincipal": "tlsconnectionmanager.amazonaws.com"
```

Datenverschlüsselung während der Übertragung in Amazon API Gateway

Die mit Amazon API Gateway erstellten APIs stellen HTTPS-Endpunkte dar. API Gateway unterstützt keine unverschlüsselten Endpunkte (HTTP).

API Gateway verwaltet die Zertifikate für `execute-api`-Standardendpunkte. Wenn Sie einen benutzerdefinierten Domainnamen konfigurieren, [geben Sie das Zertifikat für den Domainnamen](#) an. Als bewährte Methode sollten Sie keine [PIN-Zertifikate verwenden](#).

Für mehr Sicherheit können Sie eine Mindestversion des TLS-Protokolls (Transport Layer Security) auswählen, die für Ihre benutzerdefinierte API Gateway-Domäne durchgesetzt wird. WebSocket-APIs und HTTP-APIs unterstützen nur TLS 1.2. Weitere Informationen hierzu finden Sie unter [Wählen Sie eine Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway](#).

Sie können außerdem eine Amazon CloudFront-Distribution mit einem benutzerdefinierten SSL-Zertifikat in Ihrem Konto einrichten und diese mit regionalen APIs verwenden. Anschließend können Sie die Sicherheitsrichtlinie für die CloudFront-Verteilung mit TLS 1.1 oder höher entsprechend Ihren Sicherheits- und Compliance-Anfragen konfigurieren.

Weitere Informationen zum Datenschutz enthält [Schützen Sie Ihren REST APIs in API Gateway](#) und der Blog-Beitrag [AWS Shared Responsibility Model and GDPR](#) im AWS-Sicherheitsblog.

Richtlinie für den Datenverkehr zwischen Netzwerken

Mit Amazon API Gateway können Sie private REST-APIs erstellen, auf die nur von Ihrer Amazon Virtual Private Cloud (VPC) aus zugegriffen werden kann. Die VPC verwendet einen [Schnittstellen-VPC-Endpunkt](#), bei dem es sich um eine Endpunkt-Netzwerkschnittstelle handelt, die Sie in Ihrer VPC erstellen. Mithilfe von [Ressourcen-Richtlinien](#) können Sie den Zugriff auf Ihre API von aus ausgewählten VPCs und VPC-Endpunkten erlauben oder verweigern, darunter auch über AWS-Konten. Jeder Endpunkt kann für den Zugriff auf mehrere APIs verwendet werden. Sie können mit Direct Connect auch eine Verbindung von einem On-Premises-Netzwerk zu Amazon VPC herstellen und über diese Verbindung auf Ihre private API zugreifen. In allen Fällen verwendet der Datenverkehr zu Ihrer privaten API eine sichere Verbindung und verlässt nicht das Amazon-Netzwerk. Er ist vom öffentlichen Internet isoliert. Weitere Informationen hierzu finden Sie unter [the section called "Private REST APIs"](#).

Identitäts- und Zugriffsverwaltung für Amazon API Gateway

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service, den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM-Administratoren steuern, wer authentifiziert werden kann (d. h. wer sich anmelden kann) und wer autorisiert werden kann (d. h. wer Berechtigungen erhalten kann), um API Gateway-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service, das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [Funktionsweise von Amazon API Gateway mit IAM](#)
- [Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien](#)
- [Beispiele für ressourcenbasierte Richtlinien für Amazon API Gateway](#)
- [Fehlerbehebung bei Amazon API Gateway-Identität und -Zugriff](#)
- [Verwenden von serviceverknüpften Rollen für API Gateway](#)

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe [Fehlerbehebung bei Amazon API Gateway-Identität und -Zugriff](#)).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe [Funktionsweise von Amazon API Gateway mit IAM](#)).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe [Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien](#)).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter [So melden Sie sich bei Ihrem AWS-Konto an](#) im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter [AWS Signature Version 4 for API requests](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Tasks that require root user credentials](#) im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter](#) verwenden müssen.

Eine [IAM-Gruppe](#) spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#) oder indem Sie eine AWS Oder-API-Operation AWS CLI aufrufen. Weitere Informationen finden Sie unter [Methoden, um eine Rolle zu übernehmen](#) im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2

laufen. Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcen und unter welchen Bedingungen durchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter [Choose between managed policies and inline policies](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#).

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF
Weitere Informationen finden Sie unter [Übersicht über ACLs die Zugriffskontrollliste \(ACL\)](#) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
- **Richtlinien zur Dienstkontrolle (SCPs)** — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter [Service-Kontrollrichtlinien](#) im AWS Organizations -Benutzerhandbuch.
- **Richtlinien zur Ressourcenkontrolle (RCPs)** — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter [Richtlinien zur Ressourcenkontrolle \(RCPs\)](#).
- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter [Bewertungslogik für Richtlinien](#) im IAM-Benutzerhandbuch.

Funktionsweise von Amazon API Gateway mit IAM

Bevor Sie mit IAM den Zugriff auf API Gateway verwalten können, sollten Sie sich darüber informieren, welche IAM-Funktionen Sie mit API Gateway verwenden können. Einen allgemeinen Überblick darüber, wie API Gateway und andere AWS Dienste mit IAM funktionieren, finden Sie unter [AWS Services That Work with IAM](#) im IAM-Benutzerhandbuch.

Themen

- [Identitätsbasierte API-Gateway-Richtlinien](#)
- [Ressourcenbasierte API Gateway-Richtlinien](#)
- [Autorisierung basierend auf API Gateway-Tags](#)
- [API Gateway-IAM-Rollen](#)

Identitätsbasierte API-Gateway-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. API Gateway unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Weitere Informationen zu den API-Gateway-spezifischen Aktionen, Ressourcen und Bedingungsschlüsseln finden Sie im Abschnitt [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management](#) und [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management V2](#). Informationen zu allen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der [IAM-Referenz für JSON-Richtlinienelemente](#) im IAM-Benutzerhandbuch.

Das folgende Beispiel zeigt eine identitätsbasierte Richtlinie, die es einem Benutzer ermöglicht, nur private REST zu erstellen oder zu aktualisieren. APIs Weitere Beispiele finden Sie unter [the section called "Beispiele für identitätsbasierte Richtlinien"](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
```

```

    "Action": [
      "apigateway:PATCH",
      "apigateway:POST",
      "apigateway:PUT"
    ],
    "Resource": [
      "arn:aws:apigateway:us-east-1::/restapis",
      "arn:aws:apigateway:us-east-1::/restapis/?????????"
    ],
    "Condition": {
      "ForAllValues:StringEqualsIfExists": {
        "apigateway:Request/EndpointType": "PRIVATE",
        "apigateway:Resource/EndpointType": "PRIVATE"
      }
    }
  },
  {
    "Sid": "AllowResourcePolicyUpdates",
    "Effect": "Allow",
    "Action": [
      "apigateway:UpdateRestApiPolicy"
    ],
    "Resource": [
      "arn:aws:apigateway:us-east-1::/restapis/*"
    ]
  }
]
}

```

Aktionen

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie in einer Richtlinie den Zugriff erlauben oder verweigern können.

Richtlinienaktionen in API Gateway verwenden das folgende Präfix vor der Aktion: `apigateway:`. Richtlinienanweisungen müssen entweder ein `Action`- oder ein `NotAction`-Element enthalten. API Gateway definiert eine eigene Gruppe von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen können.

Der Action API-Verwaltungsausdruck hat das Format `apigateway:action`, in dem eine der folgenden API-Gateway-Aktionen `action` steht: GET, POST, PUT, DELETE, PATCH (um Ressourcen zu aktualisieren) oder *, was alle vorherigen Aktionen sind.

Einige Beispiele für den Action-Ausdruck sind:

- **apigateway:*** für alle API Gateway-Aktionen.
- **apigateway:GET** nur für die GET-Aktion in API Gateway.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [  
    "apigateway:action1",  
    "apigateway:action2"
```

Informationen zu HTTP-Verben, die für bestimmte API Gateway-Operationen verwendet werden können, finden Sie unter [Amazon API Gateway Version 1 API-Referenz](#) (REST APIs) und [Amazon API Gateway Version 2 API-Referenz](#) (WebSocket und HTTP APIs).

Weitere Informationen finden Sie unter [the section called “Beispiele für identitätsbasierte Richtlinien”](#).

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen [Amazon-Ressourcennamen \(ARN\)](#) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*" 
```

API Gateway-Ressourcen verfügen über das folgende ARN-Format:

```
arn:aws:apigateway:region::resource-path-specifier
```

Um beispielsweise eine REST-API mit der ID *api-id* und ihren Unterressourcen, wie Autorisierern in Ihrer Anweisung, anzugeben, verwenden Sie den folgenden ARN:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Um alle REST APIs - und Unterressourcen anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Eine Liste der API-Gateway-Ressourcentypen und ihrer ARNs Typen finden Sie unter [Referenz zu API Gateway Amazon-Ressourcenname \(ARN\)](#).

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die [Bedingungsoperatoren](#) verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter [Kontextschlüssel für AWS globale Bedingungen](#) im IAM-Benutzerhandbuch.

API Gateway definiert einen eigenen Satz von Bedingungsschlüsseln und unterstützt auch einige globale Bedingungsschlüssel. Eine Liste der API Gateway-Bedingungsschlüssel finden Sie unter [Bedingungsschlüssel für Amazon API Gateway](#) im IAM-Benutzerhandbuch. Informationen zu den Aktionen und Ressourcen, die Sie mit einem Bedingungsschlüssel verwenden können, finden Sie unter [Von Amazon API Gateway definierte Aktionen](#).

Informationen zur Markierung, einschließlich attributbasierter Zugriffskontrolle, finden Sie unter [Markieren](#).

Beispiele

Beispiele für identitätsbasierte API Gateway-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien](#).

Ressourcenbasierte API Gateway-Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die angeben, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für die API Gateway-Ressource ausführen kann. API Gateway unterstützt ressourcenbasierte Berechtigungsrichtlinien für REST APIs. Sie verwenden Ressourcenrichtlinien, um zu steuern, wer eine REST-API aufrufen kann. Weitere Informationen finden Sie unter [the section called “API Gateway-Ressourcenrichtlinien verwenden”](#).

Beispiele

Beispiele für ressourcenbasierte API Gateway-Richtlinien finden Sie unter [Beispiele für API Gateway-Ressourcenrichtlinien](#).

Autorisierung basierend auf API Gateway-Tags

Sie können Tags an API Gateway-Ressourcen anfügen oder Tags in einer Anforderung an API Gateway übergeben. Um den Zugriff basierend auf Tags zu steuern, stellen Sie Tag-Informationen im [Bedingungselement](#) einer Richtlinie unter Verwendung der Bedingungsschlüssel `apigateway:ResourceTag/key-name`, `aws:RequestTag/key-name` oder `aws:TagKeys` bereit. Weitere Informationen über das Markieren mit Tags von API Gateway-Ressourcen finden Sie unter [the section called “Attributbasierte Zugriffskontrolle”](#).

Beispiele für identitätsbasierte Richtlinien zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Tags dieser Ressource finden Sie unter [Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden](#).

API Gateway-IAM-Rollen

Eine [IAM-Rolle](#) ist eine Entität innerhalb Ihres AWS Kontos, die über bestimmte Berechtigungen verfügt.

Verwenden temporärer Anmeldeinformationen mit API Gateway

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API-Operationen wie [AssumeRole](#) oder [GetFederationToken](#) aufrufen.

API Gateway unterstützt die Verwendung temporärer Anmeldeinformationen.

Service-verknüpfte Rollen

Mit [dienstbezogenen Rollen](#) können AWS Dienste auf Ressourcen in anderen Diensten zugreifen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

API Gateway unterstützt serviceverknüpfte Rollen. Informationen zum Erstellen oder Verwalten von serviceverknüpften API Gateway-Rollen finden Sie unter [Verwenden von serviceverknüpften Rollen für API Gateway](#).

Servicerollen

Ein Service kann in Ihrem Namen eine [Service-Rolle](#) annehmen. Eine Service-Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Services, um eine Aktion in Ihrem Namen auszuführen. Service-Rollen werden in Ihrem IAM-Konto angezeigt und gehören dem Konto, sodass ein Administrator die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktionalität des Dienstes beeinträchtigen.

API Gateway unterstützt Servicerollen.

Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien

Standardmäßig verfügen IAM-Benutzer und -Rollen nicht über die Berechtigung zum Erstellen oder Ändern von API Gateway-Ressourcen. Sie können auch keine Aufgaben mit dem AWS-Managementkonsole AWS CLI, oder AWS SDKs ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den IAM-Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen zum Erstellen von IAM-Richtlinien finden Sie unter [Erstellen von Richtlinien im JSON-Reiter](#) im IAM-Benutzerhandbuch. Weitere Informationen zu den Aktionen, Ressourcen und Bedingungen, die für API Gateway spezifisch sind, finden Sie im Abschnitt [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management](#) und [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management V2](#).

Themen

- [Best Practices für Richtlinien](#)

- [Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer](#)
- [Einfache Leseberechtigungen](#)
- [Nur REQUEST- oder JWT-Autorisierer erstellen](#)
- [Erfordern, dass der execute-api-Standard-Endpunkt deaktiviert ist](#)
- [Erlauben Sie Benutzern, nur private REST-Dateien zu erstellen oder zu aktualisieren APIs](#)
- [Erfordern, dass API-Routen Autorisierung haben](#)
- [Verhindern Sie, dass ein Benutzer einen VPC-Link erstellt oder aktualisiert](#)
- [Beispielrichtlinien für die Verwendung von Routing-Regeln](#)

Best Practices für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand API-Gateway-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter [Von AWS verwaltete Richtlinien](#) oder [Von AWS verwaltete Richtlinien für Auftragsfunktionen](#) im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter [Richtlinien und Berechtigungen in IAM](#) im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben,

um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter [IAM-JSON-Richtlinienelemente: Bedingung](#) im IAM-Benutzerhandbuch.

- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter [Richtlinienvvalidierung mit IAM Access Analyzer](#) im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter [Sicherer API-Zugriff mit MFA](#) im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter [Best Practices für die Sicherheit in IAM](#) im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API oder. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ]
    }
  ]
}
```

```

    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
  },
  {
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
      "iam:GetGroupPolicy",
      "iam:GetPolicyVersion",
      "iam:GetPolicy",
      "iam:ListAttachedGroupPolicies",
      "iam:ListGroupPolicies",
      "iam:ListPolicyVersions",
      "iam:ListPolicies",
      "iam:ListUsers"
    ],
    "Resource": "*"
  }
]
}

```

Einfache Leseberechtigungen

Diese Beispielrichtlinie gibt einem Benutzer die Erlaubnis, Informationen über alle Ressourcen eines HTTP- oder WebSocket API-Codes mit der ID a123456789 in der AWS Region us-east-1 abzurufen. Die Ressource `arn:aws:apigateway:us-east-1::/apis/a123456789/*` umfasst alle Unterressourcen der API, wie Autorisierer und Bereitstellungen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}

```

```
]
}
```

Nur REQUEST- oder JWT-Autorisierer erstellen

[Diese Beispielrichtlinie ermöglicht es einem Benutzer, nur APIs mit REQUEST oder mit JWT Autorisierern zu erstellen, auch durch Import.](#) Im Resource-Abschnitt der Richtlinie

arn:aws:apigateway:us-east-1::/apis/??????????? ist festgelegt, dass Ressourcen maximal 10 Zeichen haben, was Unterressourcen einer API ausschließt. Dieses Beispiel verwenden `ForAllValues` im Condition-Abschnitt, da Benutzer mehrere Autorisierer gleichzeitig erstellen können, indem sie eine API importieren.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/????????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
            "REQUEST",
            "JWT"
          ]
        }
      }
    }
  ]
}
```

```
}
```

Erfordern, dass der **execute-api**-Standard-Endpunkt deaktiviert ist

Diese Beispielrichtlinie ermöglicht es Benutzern, eine API mit der Anforderung, dass `DisableExecuteApiEndpoint` `true` ist, zu erstellen, zu aktualisieren oder zu importieren. Wenn `DisableExecuteApiEndpoint` `true` ist, können Clients den `execute-api`-Standard-Endpunkt nicht zum Aufrufen einer API verwenden.

Wir verwenden die `BoolIfExists`-Bedingung, um einen Aufruf zur Aktualisierung einer API zu bearbeiten, für die der `DisableExecuteApiEndpoint`-Bedingungsschlüssel nicht ausgefüllt ist. Wenn ein Benutzer versucht, eine API zu erstellen oder zu importieren, wird der `DisableExecuteApiEndpoint`-Bedingungsschlüssel immer ausgefüllt.

Da die `apis/*` Ressource auch Unterressourcen wie Autorisierer oder Methoden erfasst, beschränken wir sie ausdrücklich nur APIs auf eine Anweisung. `Deny`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
        }
      }
    }
  ],
}
```

```
{
  "Sid": "ScopeDownToJustApis",
  "Effect": "Deny",
  "Action": [
    "apigateway:PATCH",
    "apigateway:POST",
    "apigateway:PUT"
  ],
  "Resource": [
    "arn:aws:apigateway:us-east-1::/apis/*/.*"
  ]
}
```

Erlauben Sie Benutzern, nur private REST-Dateien zu erstellen oder zu aktualisieren APIs

Diese Beispielrichtlinie verwendet Bedingungsschlüssel, um zu verlangen, dass ein Benutzer nur etwas erstellt PRIVATE APIs, und um zu verhindern, dass Updates, die eine API von PRIVATE einem anderen Typ ändern könnten, wie REGIONAL z.

Wir verwenden `ForAllValues`, um zu verlangen, dass jeder `EndpointType`, der einer API hinzugefügt wird, `PRIVATE` ist. Wir verwenden einen Ressourcenbedingungsschlüssel, um jedes Update für eine API zu ermöglichen, solange es `PRIVATE` ist. `ForAllValues` gilt nur, wenn ein Bedingungsschlüssel vorhanden ist.

Wir verwenden den Non-Greedy Matcher (?) für einen expliziten Abgleich mit der API, um zu verhindern, dass Ressourcen, die nicht IDs zur API gehören, wie Autorisierer, zugelassen werden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScopePutToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/?????????"
    ],
    "Condition": {
        "ForAllValues:StringEquals": {
            "apigateway:Resource/EndpointType": "PRIVATE"
        }
    }
},
{
    "Sid": "ScopeToPrivateApis",
    "Effect": "Allow",
    "Action": [
        "apigateway:DELETE",
        "apigateway:PATCH",
        "apigateway:POST"
    ],
    "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/?????????"
    ],
    "Condition": {
        "ForAllValues:StringEquals": {
            "apigateway:Request/EndpointType": "PRIVATE",
            "apigateway:Resource/EndpointType": "PRIVATE"
        }
    }
},
{
    "Sid": "AllowResourcePolicyUpdates",
    "Effect": "Allow",
    "Action": [
        "apigateway:UpdateRestApiPolicy"
    ],
    "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis/*"
    ]
}
]
}

```

Erfordern, dass API-Routen Autorisierung haben

Diese Richtlinie führt dazu, dass Versuche, eine Route zu erstellen oder zu aktualisieren (einschließlich durch [Import](#)), fehlschlagen, wenn die Route keine Autorisierung hat. `ForAnyValue` wird als „false“ ausgewertet, wenn der Schlüssel nicht vorhanden ist, z. B. wenn eine Route nicht erstellt oder aktualisiert wird. Wir verwenden `ForAnyValue`, da durch den Import mehrere Routen erstellt werden können.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/?????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
```



```

        "apigateway:Request/RouteAuthorizationType": "NONE"
      }
    }
  ]
}

```

Verhindern Sie, dass ein Benutzer einen VPC-Link erstellt oder aktualisiert

Diese Richtlinie verhindert, dass ein Benutzer einen VPC-Link erstellt oder aktualisiert. Ein VPC-Link ermöglicht es Ihnen, Ressourcen innerhalb einer Amazon VPC an Clients außerhalb der VPC bereitzustellen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    }
  ]
}

```

Beispielrichtlinien für die Verwendung von Routing-Regeln

Die folgenden Beispielrichtlinien zeigen, wie Sie mithilfe der RoutingRule Bedingungsschlüssel steuern können, wie Benutzer Traffic von ihren benutzerdefinierten Domainnamen zu ihrem REST

weiterleiten können. APIs Sie können diese Beispiele verwenden, um detaillierte Richtlinien zu erstellen, welche Arten von Routing-Regeln Benutzer festlegen dürfen. Weitere Informationen finden Sie unter [the section called "Routing-Regeln"](#).

Verhindern, dass ein Benutzer die Art und Weise der Weiterleitung von Anfragen durch einen benutzerdefinierten Domainnamen ändert

Diese Richtlinie verhindert, dass ein Benutzer BasePathMapping, ApiMapping oder RoutingRule erstellt oder aktualisiert. All diese Ressourcen können die Art und Weise ändern, an die ein benutzerdefinierter Domainname Anfragen weiterleitet APIs.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessBasePathMappingsApiMappingsRoutingRules",
      "Effect": "Deny",
      "Action": "apigateway:*",
      "Resource": [
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/basepathmappings/*",
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/apimappings/*",
        "arn:aws:apigateway:us-east-1:111122223333:/domainnames/example.com/routingrules/*"
      ]
    }
  ]
}
```

Benutzern erlauben, eine Routing-Regel für bestimmte Prioritäten zu aktualisieren

Diese Richtlinie erlaubt einem Benutzer nur, eine Routing-Regel auf eine Priorität zwischen 1 001 und 2 000 zu aktualisieren. Sie können diese Regel verwenden, um Ihre Produktionsregeln von Regeln mit niedrigerer Priorität zu trennen und Benutzern zu erlauben, Regeln mit niedrigerer Priorität zu ändern, ohne dass dies Auswirkungen auf die Produktionsregeln hat.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdatingRoutingRulePriorityBetween1001And2000",
      "Effect": "Allow",
      "Action": "apigateway:UpdateRoutingRule",
      "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
      "Condition": {
        "NumericGreaterThanEquals": {
          "apigateway:Resource/Priority": 1001,
          "apigateway:Request/Priority": 1001
        },
        "NumericLessThanEquals": {
          "apigateway:Resource/Priority": 2000,
          "apigateway:Request/Priority": 2000
        }
      }
    }
  ]
}
```

Benutzern erlauben, eine Routing-Regel oder eine Basispfadzuweisung für einen bestimmten Basispfadwert zu aktualisieren

Diese Richtlinie erlaubt es einem Benutzer nur, eine Basispfadzuweisung für jeden Basispfad zu aktualisieren, der mit `orders` beginnt, oder eine Routing-Regel zu aktualisieren, die zu einem Basispfad passt, der mit `orders` beginnt. Aufgrund dieser Richtlinie kann ein Benutzer eine Basispfadzuweisung oder eine Routing-Regel für `orders/create` oder `orders123` aktualisieren, aber nicht für `payment/orders`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Sid": "AllowUpdateRoutingRuleUnderPathOrders",
    "Effect": "Allow",
    "Action": "apigateway:UpdateRoutingRule",
    "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
    "Condition": {
        "ForAllValues:StringLike": {
            "apigateway:Request/ConditionBasePaths": ["orders*"],
            "apigateway:Resource/ConditionBasePaths": ["orders*"]
        },
        "Null": {
            "apigateway:Request/ConditionBasePaths": "false",
            "apigateway:Resource/ConditionBasePaths": "false"
        }
    }
}
]
}

```

Benutzern erlauben, den Routing-Modus auf bestimmte Werte zu aktualisieren

Diese Richtlinie erlaubt einem Benutzer, nur den Routing-Modus auf `API_MAPPING_ONLY` oder `ROUTING_RULE_THEN_API_MAPPING` zu aktualisieren. Weitere Informationen zum Routing-Modus finden Sie unter [the section called “Festlegen des Routing-Modus für Ihren benutzerdefinierten Domainnamen”](#).

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdateRoutingModeToAnythingWithApiMapping",
      "Effect": "Allow",
      "Action": ["apigateway:PATCH"],
      "Resource": "arn:aws:apigateway:us-east-1:./domainnames/example.com",
      "Condition": {
        "StringLike": {
          "apigateway:Request/RoutingMode": "*API_MAPPING*"
        }
      }
    }
  ]
}

```

```
}  
  ]  
}
```

Beispiele für ressourcenbasierte Richtlinien für Amazon API Gateway

Beispiele für eine ressourcenbasierte Richtlinie finden Sie unter [the section called “Beispiele für API Gateway-Ressourcenrichtlinien”](#).

Fehlerbehebung bei Amazon API Gateway-Identität und -Zugriff

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von API Gateway und IAM auftreten können.

Themen

- [Ich bin nicht zur Ausführung einer Aktion in API Gateway autorisiert](#)
- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine API-Gateway-Ressourcen ermöglichen](#)

Ich bin nicht zur Ausführung einer Aktion in API Gateway autorisiert

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer mateojackson versucht, über die Konsole Details zu einer fiktiven *my-example-widget*-Ressource anzuzeigen, jedoch nicht über `apigateway:GetWidget`-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
apigateway:GetWidget on resource: my-example-widget because no identity-based policy  
allows the GetWidget action
```

In diesem Fall muss die Richtlinie für den Benutzer mateojackson aktualisiert werden, damit er mit der `apigateway:GetWidget`-Aktion auf die *my-example-widget*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion „iam:PassRole“ autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an API Gateway übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in API Gateway auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine API-Gateway-Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob API Gateway diese Funktionen unterstützt, finden Sie unter [Funktionsweise von Amazon API Gateway mit IAM](#).
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen in AWS-Konten Ihrem Besitz gewähren können, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto, dem Sie gehören](#).

- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Verwenden von serviceverknüpften Rollen für API Gateway

Amazon API Gateway verwendet AWS Identity and Access Management (IAM) [service-verknüpfte Rollen](#). Eine serviceverknüpfte Rolle ist ein spezieller Typ einer IAM-Rolle, die direkt mit API Gateway verknüpft ist. Dienstbezogene Rollen sind von API Gateway vordefiniert und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine serviceverknüpfte Rolle vereinfacht die Einrichtung von API Gateway, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. API Gateway definiert die Berechtigungen seiner serviceverknüpften Rollen. Sofern keine andere Konfiguration festgelegt wurde, kann nur API Gateway die Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem die zugehörigen Ressourcen gelöscht wurden. Dies schützt Ihre API Gateway-Ressourcen, da Sie nicht versehentlich die Berechtigung für den Zugriff auf die Ressourcen entfernen können.

Informationen zu anderen Services, die serviceverknüpfte Rollen unterstützen, finden Sie unter [AWS -Services, die mit IAM funktionieren](#). Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Berechtigungen von serviceverknüpften Rollen für API Gateway

API Gateway verwendet die dienstbezogene Rolle mit dem Namen `AWSServiceRoleForAPIGateway`— Ermöglicht API Gateway, in Ihrem Namen auf Elastic Load Balancing, Amazon Data Firehose und andere Servicere Ressourcen zuzugreifen.

Die AWSService RoleFor APIGateway serviceverknüpfte Rolle vertraut darauf, dass die folgenden Services die Rolle übernehmen:

- ops.apigateway.amazonaws.com

Die Rollenberechtigungsrichtlinie erlaubt API Gateway die Durchführung der folgenden Aktionen für die angegebenen Ressourcen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticloadbalancing:AddListenerCertificates",
        "elasticloadbalancing:RemoveListenerCertificates",
        "elasticloadbalancing:ModifyListener",
        "elasticloadbalancing:DescribeListeners",
        "elasticloadbalancing:DescribeLoadBalancers",
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingTargets",
        "xray:GetSamplingRules",
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries",
        "servicediscovery:DiscoverInstances"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ]
    }
  ]
}
```



```

    ],
    "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "acm:DescribeCertificate",
      "acm:GetCertificate"
    ],
    "Resource": "arn:aws:acm:*:*:certificate/*"
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkInterfacePermission",
    "Resource": "arn:aws:ec2:*:*:network-interface/*"
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
      "ForAllValues:StringEquals": {
        "aws:TagKeys": [
          "Owner",
          "VpcLinkId"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:ModifyNetworkInterfaceAttribute",
      "ec2:DeleteNetworkInterface",
      "ec2:AssignPrivateIpAddresses",
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterfacePermission",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeAvailabilityZones",
      "ec2:DescribeNetworkInterfaceAttribute",
      "ec2:DescribeVpcs",
      "ec2:DescribeNetworkInterfacePermissions",
      "ec2:UnassignPrivateIpAddresses",
      "ec2:DescribeSubnets",

```

```

        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "servicediscovery:GetNamespace",
    "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
  },
  {
    "Effect": "Allow",
    "Action": "servicediscovery:GetService",
    "Resource": "arn:aws:servicediscovery:*:*:service/*"
  }
]
}

```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter [serviceverknüpfte Rollenberechtigungen](#) im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpften Rolle für API Gateway

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine API, einen benutzerdefinierten Domainnamen oder einen VPC-Link in der AWS-Managementkonsole, der oder der API erstellen AWS CLI, erstellt AWS API Gateway die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie eine API, einen benutzerdefinierten Domainnamen oder einen VPC-Link erstellen, erstellt API Gateway die serviceverknüpfte Rolle erneut für Sie.

Bearbeiten einer serviceverknüpften Rolle für API Gateway

Mit API Gateway können Sie die AWSService RoleFor APIGateway dienstverknüpfte Rolle nicht bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollename nach der Erstellung einer serviceverknüpften Rolle nicht bearbeitet werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter [Bearbeiten einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für API Gateway

Wenn Sie eine Funktion oder einen Service, die bzw. der eine serviceverknüpfte Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpfte Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

Note

Wenn der API Gateway-Service die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Um API-Gateway-Ressourcen zu löschen, die von AWSService RoleFor APIGateway

1. Öffnen Sie die API Gateway Gateway-Konsole unter <https://console.aws.amazon.com/apigateway/>.
2. Navigieren Sie zu der API, dem benutzerdefinierten Domännennamen oder dem VPC-Link, der die serviceverknüpfte Rolle verwendet.
3. Verwenden Sie die Konsole, um die Ressource zu löschen.
4. Wiederholen Sie den Vorgang APIs, um alle benutzerdefinierten Domännennamen oder VPC-Links zu löschen, die die dienstverknüpfte Rolle verwenden.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die dienstverknüpfte Rolle zu löschen. AWSService RoleFor APIGateway Weitere Informationen finden Sie unter [Löschen einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Unterstützte Regionen für serviceverknüpfte API Gateway-Rollen

API Gateway unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter [AWS -Service-Endpunkte](#).

API Gateway Gateway-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für API Gateway an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Um automatische Warnungen über Änderungen an dieser Seite erhalten, abonnieren Sie den RSS-Feed auf der API-Gateway-[Dokumentverlauf](#)-Seite.

Änderungen	Beschreibung	Datum
acm:GetCertificate - Unterstützung für AWSServiceRoleForAPIGateway - Richtlinien wurde hinzugefügt.	Die AWSServiceRoleForAPIGateway -Richtlinie enthält jetzt die Berechtigung zum Aufrufen des ACM-GetCertificate API-Aktion	12. Juli 2021
API Gateway hat mit der Verfolgung von Änderungen begonnen	API Gateway hat damit begonnen, Änderungen für seine AWS verwalteten Richtlinien zu verfolgen.	12. Juli 2021

Protokollierung und Überwachung in Amazon API Gateway

Die Überwachung ist ein wichtiger Teil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von API Gateway und Ihren AWS-Lösungen. Sie sollten Überwachungsdaten von allen Teilen Ihrer AWS-Lösung sammeln, damit Sie einen Multi-Point-Fehler leichter debuggen können, falls ein solcher auftritt. AWS bietet verschiedene Tools zur Überwachung von API Gateway und anderer Ressourcen und zur Reaktion auf potenzielle Vorfälle.

Amazon CloudWatch Logs

Um bei der Behebung von Problemen im Zusammenhang mit der Ausführung von Anfragen oder dem Clientzugriff auf Ihre API zu helfen, können Sie CloudWatch Logs aktivieren, um API-Aufrufe zu protokollieren. Weitere Informationen finden Sie unter [the section called “CloudWatch protokolliert”](#).

Amazon CloudWatch-Alarme

Mit CloudWatch-Alarmen überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird

eine Benachrichtigung an ein Thema von Amazon Simple Notification Service oder eine AWS Auto Scaling-Richtlinie gesendet. CloudWatch-Alarme lösen keine Aktionen aus, wenn sich eine Metrik in einem bestimmten Zustand befindet. Der Status muss sich stattdessen geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein. Weitere Informationen finden Sie unter [the section called “CloudWatch Metriken”](#).

Protokollierung von Zugriffen auf Firehose

Um Probleme im Zusammenhang mit Client-Zugriffen auf Ihre API zu beheben, können Sie Firehose aktivieren, um API-Aufrufe zu protokollieren. Weitere Informationen finden Sie unter [the section called “Firehose”](#).

AWS CloudTrail

CloudTrail bietet eine Aufzeichnung von Aktionen, die von einem Benutzer, einer Rolle oder einem AWS-Service in API Gateway durchgeführt wurden. Mit den von CloudTrail gesammelten Informationen können Sie die an Amazon S3 gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Angaben bestimmen. Weitere Informationen finden Sie unter [the section called “Arbeiten mit CloudTrail”](#).

AWS X-Ray

X-Ray ist ein AWS-Service, der Daten über die Anforderungen erfasst, die Ihre Anwendung verwendet, um eine Servicezuordnung zu erstellen, die Sie verwenden können, um Probleme mit Ihrer Anwendung und Möglichkeiten zur Optimierung zu identifizieren. Weitere Informationen finden Sie unter [the section called “Einrichten AWS X-Ray”](#).

AWS Config

AWS Config bietet eine detaillierte Ansicht der Konfiguration der AWS-Ressourcen in Ihrem Konto. Sie können Ressourcenbeziehungen betrachten, einen Verlauf der Konfigurationsänderungen anzeigen und feststellen, wie sich Beziehungen und Konfigurationen im Lauf der Zeit ändern. Sie können AWS Config verwenden, um Regeln zu definieren, die Ressourcenkonfigurationen im Hinblick auf Daten-Compliance auswerten. AWS Config-Regeln stellen die idealen Konfigurationseinstellungen für Ihre API-Gateway-Ressourcen dar. Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet ist, kann AWS Config Sie mit einem Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigen. Details hierzu finden Sie unter [the section called “Arbeitet mit AWS Config”](#).

Protokollieren Amazon API Gateway Gateway-API-Aufrufen mit AWS CloudTrail

Amazon API Gateway ist in einen Service integriert [AWS CloudTrail](#), der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle REST-API-Aufrufe für den API Gateway Gateway-Dienst als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der API Gateway Gateway-Konsole und Codeaufrufen an den API Gateway Gateway-Dienst APIs. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an API Gateway gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wann sie gestellt wurde, und weitere Details ermitteln.

Note

[TestInvokeAuthorizer](#) und [TestInvokeMethod](#) sind nicht angemeldet CloudTrail.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in deinem aktiv AWS-Konto , wenn du das Konto erstellst und du hast automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter [Arbeiten mit dem CloudTrail Ereignisverlauf](#). Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder [CloudTrailLake-Event-Datenspeicher](#).

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter [Erstellen eines Trails für Ihr AWS-Konto](#) und [Erstellen eines Trails für eine Organisation](#) im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter [AWS CloudTrail Preise](#). Informationen zu Amazon-S3-Preisen finden Sie unter [Amazon S3 – Preise](#).

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail [Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format](#). ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von [erweiterten Ereignisselektoren](#) auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter [Arbeiten mit AWS CloudTrail Lake](#) im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die [Preisoption](#) aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter [AWS CloudTrail Preise](#).

API Gateway Gateway-Verwaltungsereignisse in CloudTrail

[Verwaltungsereignisse](#) bieten Informationen zu Verwaltungsvorgängen, die an Ressourcen in Ihrem ausgeführt werden AWS-Konto. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. In der Standardeinstellung werden Verwaltungsereignisse CloudTrail protokolliert.

Amazon API Gateway protokolliert alle API Gateway Gateway-Aktionen als Verwaltungsereignisse, mit Ausnahme von [TestInvokeAuthorizer](#) und [TestInvokeMethod](#). Eine Liste der Amazon API Gateway-Aktionen, bei denen sich API Gateway anmeldet CloudTrail, finden Sie in der [Amazon API Gateway API-Referenz](#).

Beispiel eines API-Gateway-Ereignisses

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das die GetResource API-Gateway-Aktion demonstriert:

```
{
  Records: [
    {
      eventVersion: "1.03",
      userIdentity: {
        type: "Root",
        principalId: "AKIAI44QH8DHBEXAMPLE",
        arn: "arn:aws:iam::123456789012:root",
        accountId: "123456789012",
        accessKeyId: "AKIAIOSFODNN7EXAMPLE",
        sessionContext: {
          attributes: {
            mfaAuthenticated: "false",
            creationDate: "2015-06-16T23:37:58Z"
          }
        }
      },
      eventTime: "2015-06-17T00:47:28Z",
      eventSource: "apigateway.amazonaws.com",
      eventName: "GetResource",
      awsRegion: "us-east-1",
```



```
    sourceIpAddress: "203.0.113.11",
    userAgent: "example-user-agent-string",
    requestParameters: {
      restApiId: "3rbEXAMPLE",
      resourceId: "5tfEXAMPLE",
      template: false
    },
    responseElements: null,
    requestId: "6d9c4bfc-148a-11e5-81b6-7577cEXAMPLE",
    eventId: "4d293154-a15b-4c33-9e0a-ff5eeEXAMPLE",
    readOnly: true,
    eventType: "AwsApiCall",
    recipientAccountId: "123456789012"
  },
  ... additional entries ...
]
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter [CloudTrailDatensatzinhalte](#).

Überwachung der API-Gateway-API-Konfiguration mit AWS Config

Sie können mit [AWS Config](#) Konfigurationsänderungen an Ihren API-Gateway-API-Ressourcen aufzeichnen und Benachrichtigungen auf der Grundlage von Ressourcenänderungen senden. Die Pflege eines Konfigurations-Änderungsverlaufs für API Gateway-Ressourcen ist nützlich bei Fehlerbehebung im Betrieb sowie Anwendungsfällen wie Audit und Compliance.

AWS Config kann Änderungen verfolgen an:

- API-Stufenkonfiguration, wie z. B.:
 - Cache-Cluster-Einstellungen
 - Drosselungseinstellungen
 - Zugriffsprotokolleinstellungen
 - aktive Bereitstellung, die für die Stufe festgelegt wurde
- API-Konfiguration, wie z. B.:
 - Endpunktkonfiguration
 - Version
 - Protokoll

- Tags

Darüber hinaus können Sie AWS-Config-Regeln mit dieser Funktion Konfigurationsregeln definieren und Verstöße gegen diese Regeln automatisch erkennen, verfolgen und melden. Indem Sie Änderungen an diesen Eigenschaften der Ressourcenkonfiguration verfolgen, können Sie auch durch Änderungen ausgelöste AWS Config Regeln für Ihre API-Gateway-Ressourcen erstellen und Ihre Ressourcenkonfigurationen anhand von Best Practices testen.

Sie können die Aktivierung AWS Config in Ihrem Konto vornehmen, indem Sie die AWS Config Konsole oder die verwenden. AWS CLI Wählen Sie die Ressourcentypen aus, für die Sie Änderungen nachverfolgen möchten. Wenn Sie zuvor so konfiguriert AWS Config haben, dass alle Ressourcentypen aufgezeichnet werden, werden diese API-Gateway-Ressourcen automatisch in Ihrem Konto aufgezeichnet. Support für Amazon API Gateway in AWS Config ist in allen AWS öffentlichen Regionen und verfügbar AWS GovCloud (US). Die vollständige Liste der unterstützten Regionen finden Sie unter [Amazon-API-Gateway-Endpunkte und -Kontingente](#) in der Allgemeine AWS-Referenz.

Themen

- [Unterstützte Ressourcentypen](#)
- [Einrichten AWS Config](#)
- [Konfiguration AWS Config zum Aufzeichnen von API-Gateway-Ressourcen](#)
- [API-Gateway-Konfigurationsdetails in der AWS Config Konsole anzeigen](#)
- [Evaluieren von API-Gateway-Ressourcen mithilfe von AWS Config Regeln](#)

Unterstützte Ressourcentypen

Die folgenden API-Gateway-Ressourcentypen sind in [AWS Config Unterstützte AWS Ressourcentypen und Ressourcenbeziehungen integriert und](#) dort dokumentiert: AWS Config

- `AWS::ApiGatewayV2::Api`(WebSocket und HTTP-API)
- `AWS::ApiGateway::RestApi` (REST-API)
- `AWS::ApiGatewayV2::Stage`(WebSocket und HTTP-API-Phase)
- `AWS::ApiGateway::Stage` (REST-API-Stufe)

Weitere Informationen zu AWS Config finden Sie im [AWS Config Entwicklerhandbuch](#). Preise finden Sie auf der Seite mit [Preisinformationen zu AWS Config](#).

Important

Wenn Sie eine der folgenden API-Eigenschaften nach Bereitstellung der API ändern, müssen Sie die API [erneut bereitstellen](#), um die Änderungen zu übernehmen. Andernfalls werden die Attributänderungen in der AWS Config Konsole angezeigt, aber die vorherigen Eigenschaftseinstellungen sind weiterhin gültig. Das Laufzeitverhalten der API bleibt unverändert.

- **AWS::ApiGateway::RestApi** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
- **AWS::ApiGatewayV2::Api** – `apiKeySelectionExpression`

Einrichten AWS Config

Informationen zur ersten Einrichtung AWS Config finden Sie in den folgenden Themen im [AWS Config Entwicklerhandbuch](#).

- [Einrichtung AWS Config mit der Konsole](#)
- [Einrichtung AWS Config mit dem AWS CLI](#)

Konfiguration AWS Config zum Aufzeichnen von API-Gateway-Ressourcen

AWS Config zeichnet standardmäßig Konfigurationsänderungen für alle unterstützten Typen von regionalen Ressourcen auf, die in der Region erkannt werden, in der Ihre Umgebung ausgeführt wird. Sie können es so AWS Config anpassen, dass Änderungen nur für bestimmte Ressourcentypen oder Änderungen an globalen Ressourcen aufgezeichnet werden.

Informationen zu regionalen und globalen Ressourcen und dazu, wie Sie Ihre AWS Config Konfiguration anpassen können, finden Sie unter [Auswahl der AWS Config Ressourceneinträge](#).

API-Gateway-Konfigurationsdetails in der AWS Config Konsole anzeigen

Sie können die AWS Config Konsole verwenden, um nach API-Gateway-Ressourcen zu suchen und aktuelle und historische Details zu deren Konfigurationen abzurufen. Die folgende Prozedur zeigt, wie Sie Informationen über eine API Gateway-API finden.

So finden Sie eine API-Gateway-Ressource in der AWS Konfigurationskonsole

1. Öffnen Sie die [AWS Config -Konsole](#).
2. Wählen Sie Resources aus.
3. Wählen Sie auf der Seite Resource inventory (Ressourcenbestand) die Option Resources (Ressourcen) aus.
4. Öffnen Sie das Menü Ressourcentyp, scrollen Sie zu APIGateway oder APIGateway V2 und wählen Sie dann einen oder mehrere der API-Gateway-Ressourcentypen aus.
5. Wählen Sie Look up.
6. Wählen Sie in der angezeigten Ressourcenliste eine Ressourcen-ID aus. AWS Config AWS Config zeigt Konfigurationsdetails und andere Informationen zu der ausgewählten Ressource an.
7. Die vollständigen Details der aufgezeichneten Konfiguration können Sie mit View Details (Details anzeigen) anzeigen.

Weitere Möglichkeiten zum Auffinden einer Ressource und zum Anzeigen von Informationen auf dieser Seite finden Sie unter [AWS Ressourcenkonfigurationen und Verlauf anzeigen](#) im AWS Config Entwicklerhandbuch.

Evaluieren von API-Gateway-Ressourcen mithilfe von AWS Config Regeln

Sie können AWS Config Regeln erstellen, die die idealen Konfigurationseinstellungen für Ihre API-Gateway-Ressourcen darstellen. Sie können vordefinierte [AWS Config verwaltete Regeln](#) verwenden oder benutzerdefinierte Regeln definieren. AWS Config verfolgt kontinuierlich Änderungen an der Konfiguration Ihrer Ressourcen, um festzustellen, ob diese Änderungen gegen eine der Bedingungen in Ihren Regeln verstoßen. Die AWS Config Konsole zeigt den Konformitätsstatus Ihrer Regeln und Ressourcen an.

Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet wird, AWS Config kann Sie mithilfe eines Themas im [Amazon Simple Notification Service Developer Guide](#) (Amazon SNS) benachrichtigen. Um die Daten in diesen AWS Config Warnungen programmgesteuert zu nutzen, verwenden Sie eine Amazon-SQS-Warteschlange (Amazon Simple Queue Service) als Benachrichtigungsendpunkt für das Amazon-SNS-Thema.

Weitere Informationen zum Einrichten und Verwenden von Regeln finden Sie unter [Ressourcen mit Regeln auswerten](#) im [AWS Config -Entwicklerhandbuch](#).

Compliance-Validierung für Amazon API Gateway

Informationen darüber, ob AWS-Service ein [AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt](#), finden Sie unter [Umfang nach Compliance-Programm AWS-Services unter](#) . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter [herunterladen AWS Artifact](#). Weitere Informationen finden Sie unter [Berichte herunterladen unter](#) .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der [AWS Sicherheitsdokumentation](#).

Zuverlässigkeit in Amazon API Gateway

Im Zentrum der globalen AWS-Infrastruktur stehen die AWS-Regionen und Availability Zones (Verfügbarkeitszonen, AZs). AWS -Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über hoch redundante Netzwerke mit niedriger Latenz und hohen Durchsätzen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Als vollständig verwalteter regionaler Dienst arbeitet API Gateway in mehreren Verfügbarkeitszonen der einzelnen Regionen, wobei die Redundanz der Verfügbarkeitszonen zur Minimierung von Infrastrukturfehlern als Kategorie des Verfügbarkeitsrisikos verwendet wird. API Gateway ist so konzipiert, dass es nach dem Ausfall einer Verfügbarkeitszone automatisch wiederhergestellt wird.

Weitere Informationen über AWS-Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Um zu verhindern, dass Ihre APIs von zu vielen Anfragen überlastet werden, drosselt API Gateway Anfragen an Ihre APIs. Insbesondere setzt API Gateway ein Limit für eine Steady-State-Rate und einen Burst von Anfragen an alle APIs in Ihrem Konto. Sie können die benutzerdefinierte Drosselung für Ihre APIs konfigurieren. Weitere Informationen hierzu finden Sie unter [Drosseln Sie Anfragen an Ihren REST APIs für einen besseren Durchsatz in API Gateway](#).

Sie können Route-53-Zustandsprüfungen verwenden, um das DNS-Failover von einer API-Gateway-API in einer primären Region zu einer API-Gateway-API in einer sekundären Region zu kontrollieren. Ein Beispiel finden Sie unter [the section called “DNS-Failover”](#).

Infrastruktursicherheit in Amazon API Gateway

Als verwalteter Service ist Amazon API Gateway durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf API Gateway zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Diese API-Operationen lassen sich von einem beliebigen Netzwerkstandort aus aufrufen. Da API Gateway jedoch ressourcenbasierte Zugriffsrichtlinien unterstützt, kann es zu Einschränkungen bezüglich der Quell-IP-Adresse kommen. Sie können auch ressourcenbasierte Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (Amazon VPC) - Endpunkten oder bestimmten zu kontrollieren. VPCs Dadurch wird der Netzwerkzugriff auf eine bestimmte API-Gateway-Ressource effektiv nur von der spezifischen VPC innerhalb des AWS Netzwerks isoliert.

Schwachstellenanalyse in Amazon API Gateway

Konfiguration und IT-Steuerelemente unterliegen der übergreifenden Verantwortlichkeit von AWS und Ihnen, unserem Kunden. Weitere Informationen finden Sie unter AWS [Modell der übergreifenden Verantwortlichkeit](#).

Bewährte Methoden in Amazon API Gateway

API Gateway bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Implementieren des Zugriffs mit geringsten Berechtigungen

Verwenden Sie IAM-Richtlinien, um den Zugriff mit den geringsten Rechten für das Erstellen, Lesen, Aktualisieren oder Löschen von API Gateway APIs zu implementieren. Weitere Informationen hierzu finden Sie unter [Identitäts- und Zugriffsverwaltung für Amazon API Gateway](#). API Gateway bietet mehrere Optionen zur Steuerung des Zugriffs auf APIs die von Ihnen erstellten Daten. Weitere Informationen hierzu finden Sie unter [Steuern und verwalten Sie den Zugriff auf REST APIs in API Gateway](#), [Steuern und verwalten Sie den Zugriff WebSocket APIs auf das API Gateway](#) und [Steuern Sie den Zugriff auf HTTP APIs mit JWT-Autorisierern in API Gateway](#).

Implementieren der Protokollierung

Verwenden Sie CloudWatch Logs oder Amazon Data Firehose, um Anfragen an Ihre APIs zu protokollieren. Weitere Informationen hierzu finden Sie unter [REST APIs in API Gateway überwachen](#), [Konfigurieren Sie die Protokollierung für WebSocket APIs im API Gateway](#) und [Konfigurieren Sie die Protokollierung für HTTP APIs in API Gateway](#).

Implementieren Sie CloudWatch Amazon-Alarme

Mithilfe von CloudWatch Alarmen beobachten Sie eine einzelne Metrik über einen von Ihnen festgelegten Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird eine Benachrichtigung an ein Thema oder eine AWS Auto Scaling Richtlinie von Amazon Simple Notification Service gesendet. CloudWatch Alarme lösen keine Aktionen aus, wenn sich eine Metrik in einem bestimmten Status befindet. Der Status muss sich stattdessen geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein. Weitere Informationen finden Sie unter [the section called “CloudWatch Metriken”](#).

Aktivieren AWS CloudTrail

CloudTrail bietet eine Aufzeichnung der Aktionen, die von einem Benutzer, einer Rolle oder einem AWS Dienst in API Gateway ausgeführt wurden. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an API Gateway gestellt wurde, die IP-Adresse, von

der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln. Weitere Informationen finden Sie unter [the section called “Arbeiten mit CloudTrail”](#).

Aktivieren AWS Config

AWS Config bietet eine detaillierte Ansicht der Konfiguration der AWS Ressourcen in Ihrem Konto. Sie können Ressourcenbeziehungen betrachten, einen Verlauf der Konfigurationsänderungen anzeigen und feststellen, wie sich Beziehungen und Konfigurationen im Lauf der Zeit ändern. Sie können AWS Config damit Regeln definieren, mit denen Ressourcenkonfigurationen im Hinblick auf Datenkonformität bewertet werden. AWS Config Regeln stellen die idealen Konfigurationseinstellungen für Ihre API-Gateway-Ressourcen dar. Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet wird, AWS Config kann ich Sie mithilfe eines Amazon Simple Notification Service (Amazon SNS) - Themas benachrichtigen. Details hierzu finden Sie unter [the section called “Arbeitet mit AWS Config”](#).

Verwenden AWS Security Hub CSPM

Überwachen Sie Ihre API-Gateway-Nutzung in Bezug auf bewährte Sicherheitsmethoden, indem Sie [AWS Security Hub CSPM](#) verwenden. Security Hub CSPM verwendet Sicherheitskontrollen, um Ressourcenkonfigurationen und Sicherheitsstandards zu bewerten und Sie bei der Einhaltung verschiedener Compliance-Frameworks zu unterstützen. Weitere Informationen zur Verwendung von Security Hub CSPM zur Evaluierung von API Gateway Gateway-Ressourcen finden Sie unter [Amazon API Gateway Gateway-Kontrollen](#) im AWS Security Hub Benutzerhandbuch.

API Gateway-Ressourcen taggen

Ein Tag ist ein Metadatenlabel, das Sie zuweisen oder das einer AWS Ressource AWS zugewiesen wird. Jedes -Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment` oder `Project`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- einem optionalen Feld, dem sogenannten Tag-Wert (z. B. `111122223333` oder `Production`). Ein nicht angegebener Tag-Wert entspricht einer leeren Zeichenfolge. Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß-/Kleinschreibung unterschieden.

Tags sind für folgende Aktivitäten nützlich:

- Steuern des Zugriffs auf Ihre Ressourcen basierend auf den ihnen zugeordneten Tags. Sie können den Zugriff steuern, indem Sie in den Bedingungen für eine AWS Identity and Access Management -Richtlinie (IAM) Tag-Schlüssel und -Werte angeben. Weitere Informationen zur Tag-basierten Zugriffskontrolle finden Sie unter [Steuern des Zugriffs mit Tags](#) im IAM-Benutzerhandbuch.
- Verfolgen Sie Ihre AWS Kosten. Sie aktivieren diese Tags auf dem AWS Fakturierung und Kostenmanagement Dashboard. AWS verwendet die Tags, um Ihre Kosten zu kategorisieren und Ihnen einen monatlichen Kostenverteilungsbericht zu senden. Weitere Informationen finden Sie unter [Verwendung von Kostenzuordnungs-Tags](#) im [AWS Billing -Benutzerhandbuch](#).
- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS Dienste unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verknüpft sind. Sie könnten beispielsweise einer API-Gateway-Stufe dasselbe Tag zuweisen, das Sie einer CloudWatch Event-Regel zuweisen.

Tipps zur Verwendung von Tags finden Sie im Whitepaper [AWS -Tagging-Strategien](#).

Die folgenden Abschnitte enthalten weitere Informationen über Tags für Amazon API Gateway.

Themen

- [API Gateway-Ressourcen, die mit Tags versehen werden können](#)
- [Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden](#)

API Gateway-Ressourcen, die mit Tags versehen werden können

Tags können für die folgenden HTTP-API- oder WebSocket API-Ressourcen in der [Amazon API Gateway V2 API](#) festgelegt werden:

- Api
- DomainName
- Stage
- VpcLink

Darüber hinaus können Tags für die folgenden REST-API-Ressourcen in der [Amazon API Gateway V1-API](#) festgelegt werden:

- ApiKey
- ClientCertificate
- DomainName
- DomainNameAccessAssociation
- RestApi
- Stage
- UsagePlan
- VpcLink

Tags können nicht direkt in anderen Ressourcen festgelegt werden. In der [Amazon API Gateway V1-API](#) erben untergeordnete Ressourcen die Tags, die für übergeordneten Ressourcen festgelegt sind. Beispiel:

- Wenn ein Tag in einer RestApi-Ressource festgelegt ist, wird dieses Tag an die folgenden untergeordneten Ressourcen dieser RestApi für [Attributbasierte Zugriffskontrolle](#) vererbt:
 - Authorizer
 - Deployment
 - Documentation
 - GatewayResponse
 - Integration

- Method
 - Model
 - Resource
 - ResourcePolicy
 - Setting
 - Stage
- Wenn ein Tag für einen DomainName festgelegt ist, wird dieses Tag von allen untergeordneten BasePathMapping-, ApiMapping- und RoutingRule-Ressourcen geerbt.
 - Wenn ein Tag in einem UsagePlan festgelegt ist, wird dieses Tag von allen UsagePlanKey-Ressourcen darunter geerbt.

Note

Die Tag-Vererbung gilt nur für [attributbasierte Zugriffskontrolle](#). Sie können beispielsweise keine vererbten Tags verwenden, um die Kosten in zu überwachen AWS Cost Explorer. Wenn Sie Tags für Kostenzuweisungen verwenden möchten, empfehlen wir, Tags für untergeordnete Ressourcen zu erstellen, z. B. für die Stage-Ressource. API Gateway gibt keine geerbten Tags zurück, wenn Sie [GetTag](#) eine Ressource aufrufen.

Tag-Vererbung in der Amazon API Gateway V1-API

Zuvor war es nur möglich, Tags in Stufen festzulegen. Da Sie sie jetzt auch in anderen Ressourcen festlegen können, kann eine Stage ein Tag auf zwei Arten erhalten:

- Das Tag kann direkt in der festgelegt werde Stage.
- Die Stufe kann das Tag von der übergeordneten erbe RestApi.

Wenn eine Stufe ein Tag auf beide Arten erhält, hat das Tag den Vorrang, das direkt in der Stufe festgelegt wurde. Nehmen wir beispielsweise an, eine Stufe erbt die folgenden Tags von der übergeordneten REST-API:

```
{
  'foo': 'bar',
  'x': 'y'
```

```
}
```

Nehmen wir an, es werden auch die folgenden Tags direkt in ihr festgelegt:

```
{  
  'foo': 'bar2',  
  'hello': 'world'  
}
```

Die Wirkung für die Stufe wäre, dass sie über folgende Tags mit den folgenden Werten verfügt:

```
{  
  'foo': 'bar2',  
  'hello': 'world'  
  'x': 'y'  
}
```

Tag-Einschränkungen und Nutzungskonventionen

Die folgenden Einschränkungen und Nutzungskonventionen gelten für die Verwendung von Tags mit API Gateway-Ressourcen:

- Jede Ressource kann maximal 50 Tags haben.
- Jeder Tag muss für jede Ressource eindeutig sein. Jeder Tag kann nur einen Wert haben.
- Die maximale Länge des Tag-Schlüssels beträgt 128 Unicode-Zeichen in UTF-8.
- Die maximale Länge des Tag-Wertes beträgt 256 Unicode-Zeichen in UTF-8.
- Zulässige Zeichen für Schlüssel und Werte sind Buchstaben, Zahlen, Leerzeichen, die in UTF-8 darstellbar sind, und die folgenden Zeichen: . : + = @ _ / - (Bindestrich). Für Amazon EC2-Ressourcen können beliebige Zeichen verwendet werden.
- Bei Tag-Schlüsseln und -Werten muss die Groß-/Kleinschreibung beachtet werden. Eine bewährte Methode besteht darin, sich für eine einheitliche Schreibweise der Tag-Benennungen zu entscheiden und diese Strategie für alle Ressourcentypen umzusetzen. Entscheiden Sie sich beispielsweise für `Costcenter`, `costcenter` oder `CostCenter` und verwenden Sie diese Konvention für alle Tags. Vermeiden Sie die Verwendung von ähnlichen Tags mit uneinheitlicher Fallunterscheidung.
- Das Präfix `aws:` darf nicht für Tags verwendet werden; es ist für die AWS -Verwendung reserviert. Sie können keine Tag-Schlüssel oder -Werte mit diesem Präfix bearbeiten oder löschen. Tags mit diesem Präfix werden nicht als Ihre Tags pro Ressourcenlimit angerechnet.

Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden

Bedingungen in AWS Identity and Access Management Richtlinien sind Teil der Syntax, mit der Sie Berechtigungen für API-Gateway-Ressourcen angeben. Einzelheiten zur Angabe von IAM-Richtlinien finden Sie unter [the section called “Verwenden von IAM-Berechtigungen”](#). In API Gateway können Ressourcen Tags haben. Einige Aktionen können ebenfalls Tags enthalten. Wenn Sie eine IAM-Richtlinie erstellen, können Sie Tag-Bedingungsschlüssel verwenden, um Folgendes zu kontrollieren:

- Welche Benutzer auf der Grundlage von Tags, die die Ressource bereits besitzt, Aktionen mit einer API Gateway-Ressource ausführen können.
- Welche Tags in der Anforderung einer Aktion übergeben werden können.
- Ob bestimmte Tag-Schlüssel in einer Anforderung verwendet werden können.

Die Verwendung von Tags für die attributbasierte Zugriffskontrolle kann eine feinere Kontrolle als die Kontrolle auf API-Ebene sowie eine dynamischere Kontrolle als die ressourcenbasierte Zugriffskontrolle ermöglichen. IAM-Richtlinien können erstellt werden, die eine Operation basierend auf Tags in der Anforderung (Anforderungs-Tags) oder Tags in der Ressource, mit der gearbeitet wird (Ressourcen-Tags), zulassen oder ablehnen. Im Allgemeinen sind Ressourcen-Tags für Ressourcen vorgesehen, die bereits vorhanden ist. Anforderungs-Tags sind dafür da, wenn Sie neue Ressourcen erstellen.

Die vollständige Syntax und Semantik der Tag-Bedingungsschlüssel finden Sie unter [Steuern des Zugriffs mit Tags](#) im IAM-Benutzerhandbuch.

Die folgenden Beispiele zeigen, wie Tag-Bedingungen in Richtlinien für API Gateway-Benutzer angegeben werden können.

Einschränken von Aktionen basierend auf Ressourcen-Tags

Die folgende Beispielrichtlinie erteilt Benutzern die Berechtigung zum Ausführen aller Aktionen für alle Ressourcen, sofern diese Ressourcen nicht über das Tag „Environment“ mit dem Wert „prod“ verfügen.

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "apigateway:*",
    "Resource": "*"
  },
  {
    "Effect": "Deny",
    "Action": [
      "apigateway:*"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Environment": "prod"
      }
    }
  }
]
}

```

Zulassen von Aktionen basierend auf Ressourcen-Tags

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle Aktionen für API Gateway-Ressourcen auszuführen, sofern die Ressourcen über das Tag „Environment“ mit dem Wert „Development“ verfügen. Die Deny-Anweisung verhindert, dass der Benutzer den Wert des Environment-Tags ändert.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConditionallyAllow",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [

```

```

        "arn:aws:apigateway:*:*:*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Environment": "Development"
        }
    }
},
{
    "Sid": "AllowTagging",
    "Effect": "Allow",
    "Action": [
        "apigateway:*"
    ],
    "Resource": [
        "arn:aws:apigateway:*:*:/tags/*"
    ]
},
{
    "Sid": "DenyChangingTag",
    "Effect": "Deny",
    "Action": [
        "apigateway:*"
    ],
    "Resource": [
        "arn:aws:apigateway:*:*:/tags/*"
    ],
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:TagKeys": "Environment"
        }
    }
}
]
}

```

Tagging-Operationen verweigern

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle API Gateway-Aktionen auszuführen, mit Ausnahme des Änderns von Tags.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "arn:aws:apigateway:*::/tags*"
    }
  ]
}
```

Tagging-Operationen erlauben

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle API Gateway-Ressourcen abzurufen und die Tags für diese Ressourcen zu ändern. Um die Tags für eine Ressource abzurufen, muss der Benutzer über GET-Berechtigungen für diese Ressource verfügen. Um die Tags für eine Ressource zu aktualisieren, muss der Benutzer über PATCH-Berechtigungen für diese Ressource verfügen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
```



```
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:DELETE"
    ],
    "Resource": [
        "arn:aws:apigateway:*::/tags/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "apigateway:GET",
        "apigateway:PATCH"
    ],
    "Resource": [
        "arn:aws:apigateway:*::*"
    ]
}
]
```

API-Referenzen

Amazon API Gateway ermöglicht APIs die Erstellung und Bereitstellung Ihres eigenen HTTP- und WebSocket APIs. Darüber hinaus APIs sind API Gateway standardmäßig verfügbar AWS SDKs.

Wenn Sie eine Sprache verwenden, für die es ein AWS SDK gibt, ziehen Sie es möglicherweise vor, das SDK zu verwenden, anstatt das API-Gateway-REST APIs direkt zu verwenden. SDKs Sie vereinfachen die Authentifizierung, lassen sich problemlos in Ihre Entwicklungsumgebung integrieren und bieten einfachen Zugriff auf API Gateway Gateway-Befehle.

Hier finden Sie die REST-API-Referenzdokumentation AWS SDKs und die API Gateway REST-API-Referenzdokumentation:

- [Tools für Amazon Web Services](#)
- [REST-API-Referenz für Amazon API Gateway](#)
- [Amazon API Gateway WebSocket und HTTP-API-Referenz](#)

Service-Endpunkte in API Gateway

Ein Endpunkt ist eine URL, die als Einstiegspunkt für einen AWS Webservice dient. API Gateway unterstützt die folgenden Endpunkttypen:

- [the section called “IPv4 Endpunkte”](#)
- [Dualstack-Endpunkte](#) (unterstützen sowohl als auch) IPv4 IPv6
- [FIPS-Endpunkte](#)

Wenn Sie eine Anfrage stellen, können Sie den zu verwendenden Endpunkt angeben. Wenn Sie keinen Endpunkt angeben, wird der IPv4 Endpunkt standardmäßig verwendet. Um einen anderen Endpunkttyp zu verwenden, müssen Sie ihn in Ihrer Anforderung angeben. Beispiele für diese Vorgehensweise finden Sie unter [the section called “Angaben von Endpunkten”](#). Eine Tabelle mit den verfügbaren Endpunkten finden Sie unter [Endpunkte von Amazon API Gateway](#).

IPv4 Endpunkte

IPv4 Endpunkte unterstützen nur IPv4 Datenverkehr. IPv4 Endpunkte sind für alle Regionen verfügbar.

Wenn Sie den allgemeinen Endpunkt, `apigateway.amazonaws.com`, angeben, verwenden wir den Endpunkt für `us-east-1`. Um eine andere Region zu verwenden, geben Sie den zugehörigen Endpunkt an. Wenn Sie beispielsweise `apigateway.us-east-2.amazonaws.com` als Endpunkt angeben, leiten wir Ihre Anfrage an den `us-east-2`-Endpunkt weiter.

IPv4 Endpunktnamen verwenden die folgende Benennungskonvention:

- `apigateway.region.amazonaws.com`

Der IPv4 Endpunktname für die `eu-west-1` Region lautet beispielsweise `apigateway.eu-west-1.amazonaws.com`.

Dualstack IPv4 - (und) Endpunkte IPv6

Dualstack-Endpunkte unterstützen sowohl den als auch den Datenverkehr. IPv4 IPv6 Wenn Sie eine Anfrage an einen Dual-Stack-Endpunkt stellen, wird die Endpunkt-URL je nach dem von Ihrem Netzwerk und Client verwendeten Protokoll in eine IPv6 oder eine IPv4 Adresse aufgelöst.

Dual-Stack-Endpunktnamen verwenden die folgende Namenskonvention:

- `apigateway.region.api.aws`

Beispielsweise ist der Dual-Stack-Endpunktname für die Region `eu-west-1` `apigateway.eu-west-1.api.aws`.

Angeben von Endpunkten

Die folgenden Beispiele zeigen, wie Sie mithilfe von `aws cli` einen Endpunkt für die `us-east-2` Region angeben. AWS CLI `apigateway`

- Dualstack

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

Die folgenden Beispiele zeigen, wie Sie mithilfe von für einen Endpunkt AWS CLI für die us-east-2 Region angeben `apigatewayv2`.

- Dualstack

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

IPv6 Adressen in IAM-Richtlinien verwenden

Wenn Sie IAM-Benutzerrichtlinien oder API Gateway-Ressourcenrichtlinien verwenden, um den Zugriff auf API Gateway oder ein API Gateway zu steuern APIs, stellen Sie sicher, dass Ihre Richtlinien aktualisiert wurden und IPv6 Adressbereiche enthalten. Richtlinien, die nicht für den Umgang mit IPv6 Adressen aktualisiert wurden, können sich auf den Zugriff von Clients auf API Gateway auswirken, wenn sie den Dualstack-Endpunkt verwenden. Weitere Informationen über die Verwaltung von Zugriffsberechtigungen mit IAM finden Sie unter [Identitäts- und Zugriffsverwaltung für Amazon API Gateway](#).

IAM-Richtlinien, die IP-Adressen filtern, verwenden [Bedingungsoperatoren für IP-Adressen](#).

Die folgende Identitätsrichtlinie ermöglicht es IP-Adressen in diesem `54.240.143.*` Bereich, Informationen über alle Ressourcen eines HTTP- oder WebSocket APIs mit der Kennung von abzurufen. `a123456789` Alle IP-Adressen außerhalb dieses Bereichs werden beim Zugriff auf Informationen zu den Ressourcen der API abgelehnt. Da alle IPv6 Adressen außerhalb des zulässigen Bereichs liegen, verhindert diese Richtlinie, dass IPv6 Adressen auf Informationen über die API zugreifen können.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
```

```
"Effect": "Allow",
"Principal": "*",
"Action": "apigateway:GET",
"Resource": "arn:aws:apigateway:us-east-1::/apis/a123456789/*",
"Condition": {
  "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
}
]
```

Sie können das Condition Element der API-Richtlinie so ändern, dass sowohl IPv4 (54.240.143.0/24) als auch IPv6 (2001:DB8:1234:5678::/64) Adressbereiche zulässig sind, wie im folgenden Beispiel gezeigt. Sie können denselben Typ von Condition-Block, der im Beispiel gezeigt wird, verwenden, um sowohl Ihre IAM-Benutzerrichtlinien als auch die API-Gateway-Ressourcensicherheitsrichtlinien zu aktualisieren.

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

Kontingente für Amazon API Gateway

Die folgenden Kontingente gelten für alle API-Typen von Amazon API Gateway.

API Gateway-Kontingent auf Kontoebene, pro Region

Die folgenden Kontingente gelten pro Konto und pro Region in Amazon API Gateway.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Drosseln Sie das Kontingent pro Konto APIs, pro Region für HTTP APIs WebSocket APIs, REST und WebSocket Callback APIs	10 000 Anforderungen pro Sekunde (RPS) mit einer zusätzlichen Steigerungskapazität über den Token-Bucket-Algorithmus, unter Verwendung einer maximalen Bucket-Kapazität von 5 000 Anforderungen. *  Note Das Burst-Kontingent wird vom API Gateway Service-Team auf der Grundlage des Gesamt-RPS-Kontingents für das Konto in der Region festgelegt. Kunden können dieses Kontingent nicht kontrollieren und keine Änderungen daran anfordern.	Ja
Drosseln Sie das Kontingent ohne Zugriffskontrolle pro Konto und Region für ein Portal	250.000 Anfragen pro Sekunde	Nein
Drosselung des Kontingents mit Zugriffskontrolle pro Konto und	10.000 Anfragen pro Sekunde	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden

Region für ein Portal

* Für die folgenden Regionen beträgt das Standard-Drosselungskontingent 2 500 RPS und das Standard-Burst-Kontingent 1 250 RPS: Afrika (Kapstadt), Europa (Mailand), Asien-Pazifik (Jakarta), Naher Osten (VAE), Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv), Kanada West (Calgary), Asien-Pazifik (Malaysia), Asien-Pazifik (Thailand) und Mexiko (Zentral).

API Gateway-Kontingente für das Erstellen, Bereitstellen und Verwalten einer API

Die folgenden festen Kontingente gelten für das Erstellen, Bereitstellen und Verwalten einer API in API Gateway mithilfe der AWS CLI API Gateway Gateway-Konsole oder der API Gateway Gateway-REST-API und ihrer SDKs. Diese Kontingente können nicht erhöht werden.

Action	Standardkontingent	Kann erhöht werden
CreateApiKey	5 Anforderungen pro Sekunde pro Konto	Nein
CreateDeployment	1 Anforderung alle 5 Sekunden pro Konto	Nein
CreateDocumentationVersion	1 Anforderung alle 20 Sekunden pro Konto	Nein
CreateDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
CreateResource	5 Anforderungen pro Sekunde pro Konto	Nein

Action	Standardkontingent	Kann erhöht werden
CreateRestApi für regionale oder private APIs	1 Anforderung alle 3 Sekunden pro Konto	Nein
CreateRestApi für Edge-optimierte API	1 Anforderung alle 30 Sekunden pro Konto	Nein
CreateVpcLink (V2)	1 Anforderung alle 15 Sekunden pro Konto	Nein
DeleteApiKey	5 Anforderungen pro Sekunde pro Konto	Nein
DeleteDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
DeleteResource	5 Anforderungen pro Sekunde pro Konto	Nein
DeleteRestApi	1 Anforderung alle 30 Sekunden pro Konto	Nein
GetResources	5 Anforderungen alle 2 Sekunden pro Konto	Nein
DeleteVpcLink (V2)	1 Anforderung alle 30 Sekunden pro Konto	Nein
ImportDocumentationParts	1 Anforderung alle 30 Sekunden pro Konto	Nein
ImportRestApi für regionale oder private API	1 Anforderung alle 3 Sekunden pro Konto	Nein
ImportRestApi für Edge-optimierte API	1 Anforderung alle 30 Sekunden pro Konto	Nein
PutRestApi	1 Anforderung pro Sekunde pro Konto	Nein

Action	Standardkontingent	Kann erhöht werden
UpdateAccount	1 Anforderung alle 20 Sekunden pro Konto	Nein
UpdateDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
UpdateUsagePlan	1 Anforderung alle 20 Sekunden pro Konto	Nein
Portal erstellen	1 Anfrage alle 3 Sekunden	Nein
Portal aktualisieren	2 Anfragen pro Minute	Nein
Holen Sie sich das Portal	10 Anforderungen pro Sekunde	Nein
Portale auflisten	10 Anforderungen pro Sekunde	Nein
Portal veröffentlichen	2 Anfragen pro Minute	Nein
DeletePortal	2 Anfragen pro Minute	Nein
PreviewPortal	1 Anfrage alle 3 Sekunden	Nein
DisablePortal	2 Anfragen pro Minute	Nein
GetPortalProduct	10 Anforderungen pro Sekunde	Nein
ListPortalProduct	5 Anforderungen pro Sekunde	Nein
CreatePortalProduct	2 Anfragen pro Sekunde	Nein
UpdatePortalProduct	0,5 Anfragen pro Sekunde	Nein
DeletePortalProduct	1 Anfrage pro Sekunde	Nein
PutPortalProductSharingPolicy	1 Anfrage alle 3 Sekunden	Nein

Action	Standardkontingent	Kann erhöht werden
GetPortalProductSharingPolicy	10 Anforderungen pro Sekunde	Nein
DeletePortalProductSharingPolicy	1 Anfrage alle 3 Sekunden	Nein
CreateProductRestEndpointPage	1 Anfrage alle 3 Sekunden	Nein
GetProductRestEndpointPage	10 Anforderungen pro Sekunde	Nein
UpdateProductRestEndpointPage	1 Anfrage alle 3 Sekunden	Nein
DeleteProductRestEndpointPage	1 Anfrage alle 3 Sekunden	Nein
ListProductRestEndpointPages	10 Anforderungen pro Sekunde	Nein
CreateProductPage	1 Anfrage alle 3 Sekunden	Nein
GetProductPage	10 Anforderungen pro Sekunde	Nein
UpdateProductPage	1 Anfrage alle 3 Sekunden	Nein
DeleteProductPage	1 Anfrage alle 3 Sekunden	Nein
ListProductPages	10 Anforderungen pro Sekunde	Nein
Andere Operationen	Kein Kontingent bis zum Gesamtkontokontingent.	Nein

Action	Standardkontingent	Kann erhöht werden
Gesamte Operationen	10 Anforderungen pro Sekunde mit einem Burst-Kontingent von 40 Anforderungen pro Sekunde.	Nein

Kontingente für die Konfiguration und Ausführung einer REST-API in API Gateway

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer REST-API in Amazon API Gateway. Wenn das Kontingent pro API gilt, kann es nur für bestimmte Konten erhöht werden APIs, nicht für alle. APIs

Weitere Informationen zu Kontingenten auf Kontoebene finden Sie unter [Kontingente und wichtige Hinweise](#).

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Öffentliche benutzerdefinierte Domainnamen pro Konto pro Region	120	Ja
Private benutzerdefinierte Domainnamen pro Konto pro Region	50	Ja
Regional APIs pro Region	600	Nein
Edge-optimiert pro Region APIs	120	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Privat APIs pro Konto pro Region	600	Nein
Domainnamen zugriffszuweisungen pro Konto	100	Ja
API-Zuordnungen auf mehreren Ebenen pro Domain	200	Nein
Routing-Regeln pro Domain	50	Ja
Länge (in Zeichen) der URL für eine Edge-optimierte API	8192	Nein
Länge der URL für eine regionale API in Zeichen	10240	Nein
Länge der URL für eine private API in Zeichen	8192	Nein
Länge in Zeichen der API Gateway-Ressourcenrichtlinie	8192	Ja
API-Schlüssel pro Konto pro Region	10000	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Clientzertifikate pro Konto pro Region	60	Ja
Autorisierer pro API (AWS Lambda und Amazon Cognito)	10	Ja Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.
Dokumentationsteile pro API	2000	Ja Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.
Ressourcen pro API	300	Ja

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Stufen pro API	10	Ja
Stufenvariablen pro Stufe	100	Nein
Länge (in Zeichen) des Schlüssels in einer Stufenvariable	64	Nein
Länge (in Zeichen) des Werts in einer Stufenvariable	512	Nein
Nutzungspläne pro Konto pro Region	300	Ja
Nutzungspläne pro API-Schlüssel	10	Ja
VPC-Links pro Konto pro Region	20	Ja
API-Caching-TTL	300 Sekunden standardmäßig und konfigurierbar zwischen 0 und 3.600 durch einen API-Eigentümer.	Nicht für die obere Grenze (3.600)
Zwischengespeicherte Antwortgröße	1 048 576 Bytes Durch Datenverschlüsselung kann sich die Größe des zwischengespeicherten Elements erhöhen.	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Timeout bei der Integration für Regional APIs	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Ja*
Timeout für die Integration bei Edge-Optimized APIs	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Nein
Timeout für die Integration im privaten Bereich APIs	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Ja*
Gesamtgröße aller Header-Werte, einschließlich Header-Namen, Werte, Zeilenabschlusszeichen und Leerzeichen	10 240 Byte	Nein
Gesamte kombinierte Größe aller Header-Werte für eine private API	8 000 Byte	Nein
Nutzlastgröße	10 MB	Nein
Tags pro Stufe	50	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Anzahl der Iterationen in einer <code>#foreach ... #end</code> -Schleife in Zuordnungsvorlagen	1000	Nein
Maximale Größe der Zuweisungsvorlage	300 KB	Nein
Größe der Zugriffsprotokollvorlage	3 KB	Nein
Länge des Methoden-ARN	1 600 Bytes Wenn Ihre Methode einen Pfadparameter enthält und ein Client einen Wert verwendet, der die ARN-Länge überschreitet, gibt Ihre API die Antwort <code>414 Request URI too long</code> zurück.	Nein
Ablehnungseinstellungen auf Methodenebene für eine Phase in einem Nutzungsplan.	20	Ja Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Modellgröße pro API	400 KB	Nein
Anzahl der Zertifikate in einem Truststore	1.000 Zertifikate mit einer Gesamtobjektgröße bis 1 MB.	Nein
Leerlaufverbindungszeitlimit	310 Sekunden	Nein
Maximale Größe der API-Definitionsdatei bei Verwendung von restapi:import oder restapi:put	6 MB	Nein

* Das Integrations-Timeout kann nicht auf weniger als 50 Millisekunden gesetzt werden. Sie können die Zeitüberschreitung für die Integration auf mehr als 29 Sekunden erhöhen. Jedoch kann dies eine Reduzierung des Drosselungskontingents für das Konto auf regionaler Ebene erfordern.

Bewährte Methoden zur Reduzierung Ihrer Kontingente

Die folgenden bewährten Methoden können Ihnen helfen, Ihre aktuelle Anzahl an Ressourcen zu reduzieren, um eine Erhöhung Ihres Kontingents zu vermeiden. Stellen Sie sicher, dass diese Vorschläge zur Architektur Ihrer API passen.

APIs pro Region

Um die Anzahl der APIs pro Region zu reduzieren, exportieren Sie alle ungenutzten APIs und löschen Sie sie dann aus API Gateway. Weitere Informationen finden Sie unter [the section called “Exportieren einer REST-API”](#).

Stufen pro API

Um die Anzahl der Stufen pro API zu reduzieren, teilen Sie Ihre API in mehrere auf APIs.

Ressourcen pro API

Verwenden Sie {proxy+}-Pfade, um die Anzahl der Ressourcen zu reduzieren. Weitere Informationen finden Sie unter [the section called “Einrichten von Proxy-Integrationen mit einer Proxy-Ressource”](#).

API-Zuweisungen

Um die Anzahl der API-Zuweisungen für einen benutzerdefinierten Domainnamen zu verringern, verwenden Sie API-Zuweisungen mit nur einer Ebene, z. B. /prod. API-Zuweisungen mit nur einer Ebene werden nicht auf das Kontingent für API-Zuweisungen angerechnet.

Genehmiger pro API

Um die Anzahl der Genehmiger pro API zu reduzieren, können Sie sie für verschiedene API-Methoden wiederverwenden.

Dokumentationsteile pro API

Um die Anzahl der Dokumentationsteile pro API zu reduzieren, verwenden Sie `ignore=documentation`, wenn Sie Ihre API importieren. Weitere Informationen finden Sie unter [the section called “Importieren einer API-Dokumentation”](#).

Sie können außerdem die Vererbung von Inhalten verwenden, damit einige Dokumentationsteile Inhalte von einer API-Entität mit allgemeineren Spezifikationen übernehmen. Weitere Informationen finden Sie unter [the section called “Darstellung der API-Dokumentation in API Gateway”](#).

Länge in Zeichen der API Gateway-Ressourcenrichtlinie

Um die Länge einer Ressourcenrichtlinie zu reduzieren, verwenden Sie AWS WAFV2 zum Schutz Ihrer API. Weitere Informationen finden Sie unter [the section called “AWS WAF”](#).

Wenn Ihre Richtlinie IP-Adressen enthält, können Sie auch Bereiche anstelle einzelner Werte zur Definition von IP-Adressen verwenden.

Nutzungspläne pro API-Schlüssel

Um die Anzahl der Nutzungspläne pro API-Schlüssel zu reduzieren, verwenden Sie einen API-Schlüssel pro Nutzungsplan und verknüpfen Sie Ihren Nutzungsplan mit mehreren APIs. Wir empfehlen nicht, einen API-Schlüssel für mehrere Nutzungspläne gemeinsam zu nutzen.

Kontingente für Konfiguration und Ausführung einer HTTP-API in API Gateway

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer HTTP-API in Amazon API Gateway. Wenn das Kontingent pro API gilt, kann es nur für bestimmte Konten erhöht werden APIs, nicht für alle. APIs Weitere Informationen zu Kontingenten auf Kontoebene finden Sie unter [Kontingente und wichtige Hinweise](#).

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Routen pro API	300	Ja
Integrationen pro API	300	Nein
Maximales Timeout für Integrationen	30 Sekunden	Nein
Stufen pro API	10	Ja
API-Zuweisungen auf mehreren Ebenen pro Domain	200 API-Zuweisungen Dieses Limit beinhaltet keine API-Zuweisungen mit einer einzigen Ebene, wie z. B. /prod.	Nein
Tags pro Stufe	50	Nein
Gesamte kombinierte Größe der Anfragezeilen- und Header-Werte	10 240 Byte	Nein
Nutzlastgröße	10 MB	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Benutzerdefinierte Domänen pro Konto pro Region	120	Ja
Größe der Zugriffsprotokollvorlage	3 KB	Nein
Amazon CloudWatch Logs-Protokolleintrag	1 MB	Nein
Genehmiger pro API	10	Ja Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.
Zielgruppen pro Genehmiger	50	Nein
Bereiche pro Route	10	Nein
Timeout für den JSON Web Key Set-Endpunkt	1.500 ms	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Antwortgröße des JSON Web Key Set-Endpunkts	150 000 Byte	Nein
Timeout für OpenID Connect-Erkennungsendpunkt	1.500 ms	Nein
Timeout der Antwort des Lambda-Funktionshandlers	10000 ms	Nein
VPC-Links pro Konto pro Region	10	Ja
Subnetze pro VPC-Link	10	Ja
Stufenvariablen pro Stufe	100	Nein
Länge (in Zeichen) des Schlüssels in einer Stufenvariable	64	Nein
Länge (in Zeichen) des Werts in einer Stufenvariable	512	Nein

Bewährte Methoden zur Reduzierung Ihrer Kontingente

Die folgenden bewährten Methoden können Ihnen helfen, Ihre aktuelle Anzahl an Ressourcen zu reduzieren, um eine Erhöhung Ihres Kontingents zu vermeiden. Stellen Sie sicher, dass diese Vorschläge zur Architektur Ihrer API passen.

APIs pro Region

Um die Anzahl der APIs pro Region zu reduzieren, exportieren Sie alle ungenutzten APIs und löschen Sie sie dann aus API Gateway. Weitere Informationen finden Sie unter [the section called “Exportieren”](#).

Stufen pro API

Um die Anzahl der Stufen pro API zu reduzieren, teilen Sie Ihre API in mehrere auf APIs.

Genehmiger pro API

Um die Anzahl der Genehmiger pro API zu reduzieren, können Sie sie für verschiedene API-Methoden wiederverwenden.

Kontingente für die Konfiguration von Portalen in API Gateway

Die folgenden Kontingente gelten für die Erstellung von Portalen in API Gateway. Weitere Informationen finden Sie unter [API-Gateway-Portale](#).

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Portale pro Konto	15	Ja
Portalprodukte pro Portal	200	Nein
Portalprodukte pro Konto	500	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Produkt-REST-Endpunktseiten pro Portalprodukt	40	Ja
Produktseiten pro Portalprodukt	40	Ja
Größe des Logos	200 KB	Nein
Seitengröße der Dokumentation pro REST-Endpunktseite des Produkts	32.000 Zeichen	Nein
Benutzerdefiniertes Seitenformat für Produktseiten	32.000 Zeichen	Nein
Benutzerdefinierte Domainnamen pro Portal	1	Nein
Autorisierer pro Portal	1	Nein

Kontingente für die Konfiguration und Ausführung eines WebSocket integrierten API Gateway

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer WebSocket API in Amazon API Gateway. Wenn das Kontingent pro API gilt, kann es nur für ein bestimmtes Konto erhöht werden APIs, nicht für alle APIs in einem Konto. Weitere Informationen zu Kontingenten auf Kontoebene finden Sie unter [Kontingente und wichtige Hinweise](#).

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Neue Verbindungen pro Sekunde pro Konto (über alle WebSocket APIs) pro Region	500	Ja
Gleichzeitige Verbindungen	Nicht zutreffend *	Nicht zutreffend
AWS Lambda Autorisierer pro API	10	Ja Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.
AWS Lambda Größe des Autorisierer-Ergebnisses	8 KB	Nein
Routen pro API	300	Ja
Integrationen pro API	300	Ja

Ressource oder Operation	Standardkontingent	Kann erhöht werden
		Wenden Sie sich an das AWS - Supportcenter , um dieses Kontingent zu erhöhen.
Timeout bei der Integration	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Nein
Stufen pro API	10	Ja
WebSocket Größe des Frames	32 KB	Nein
Nachrichten-Nutzlastgröße	128 KB *	Nein
Maximale Größe der Zuweisungsvorlage	300 KB	Nein
Verbindungsdauer für die WebSocket API	2 Stunden	Nein
Zeitlimit für Verbindungsleerlauf	10 Minuten	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Länge der URL für eine WebSocket API in Zeichen	4096	Nein
Größe der Zugriffsprotokollvorlage	3 KB	Nein

* API Gateway erzwingt kein Kontingent für gleichzeitige Verbindungen. Die maximale Anzahl gleichzeitiger Verbindungen wird durch die Rate neuer Verbindungen pro Sekunde und die maximale Verbindungsdauer von zwei Stunden bestimmt. Wenn Clients beispielsweise mit dem Standardkontingent von 500 neuen Verbindungen pro Sekunde eine Verbindung mit der maximalen Rate über zwei Stunden herstellen, kann API Gateway bis zu 3 600 000 gleichzeitige Verbindungen bedienen.

** Aufgrund des Kontingents für die WebSocket Framegröße von 32 KB muss eine Nachricht, die größer als 32 KB ist, in mehrere Frames aufgeteilt werden, die jeweils 32 KB oder kleiner sind. Dies gilt für `@connections`-Befehle. Wenn eine größere Nachricht (oder eine größere Frame-Größe) empfangen wird, wird die Verbindung mit dem Code 1009 geschlossen.

Wichtige Hinweise zu Amazon API Gateway

Im folgenden Abschnitt finden Sie Hinweise, die sich auf Ihre Verwendung von API Gateway auswirken könnten.

Themen

- [Amazon API Gateway: Wichtige Hinweise für HTTP APIs](#)
- [Amazon API Gateway wichtige Hinweise für HTTP und WebSocket APIs](#)
- [Amazon API Gateway wichtige Hinweise für REST und WebSocket APIs](#)
- [Amazon API Gateway wichtige Hinweise für WebSocket APIs](#)
- [Amazon API Gateway — wichtige Hinweise für REST APIs](#)

Amazon API Gateway: Wichtige Hinweise für HTTP APIs

- HTTP APIs übersetzt eingehende X-Forwarded-* Header in einen Forwarded Standard-Header und fügt die Ausgangs-IP, den Host und das Protokoll an.
- API Gateway fügt Ihrer Anfrage den Content-Type-Header hinzu, wenn keine Nutzdaten vorhanden sind und die Content-Length 0 ist.

Amazon API Gateway wichtige Hinweise für HTTP und WebSocket APIs

- Signature Version 4A wird von Amazon API Gateway für HTTP und nicht offiziell unterstützt WebSocket APIs.

Amazon API Gateway wichtige Hinweise für REST und WebSocket APIs

- API Gateway unterstützt nicht die gemeinsame Nutzung eines benutzerdefinierten Domainnamens über REST und WebSocket APIs.
- Phasennamen dürfen nur alphanumerische Zeichen sowie Binde- und Unterstriche enthalten. Die maximale Länge beträgt 128 Zeichen.
- Die Pfade /ping und /sping sind für die Servicezustandsprüfung reserviert. Wenn diese für API-Ressourcen auf Stammebene mit benutzerdefinierten Domains verwendet werden, liefern sie nicht das erwartete Ergebnis.
- API Gateway begrenzt Protokollereignisse derzeit auf 1024 Bytes. Protokollereignisse, die größer als 1024 Byte sind, wie Anforderungs- und Antworttexte, werden von API Gateway vor der Übermittlung an CloudWatch Logs gekürzt.
- CloudWatch Metrics begrenzt derzeit Dimensionsnamen und -werte auf 255 gültige XML-Zeichen. (Weitere Informationen finden Sie im [CloudWatch Benutzerhandbuch](#).) Dimensionenwerte sind eine Funktion benutzerdefinierter Namen, einschließlich API-Name, Bezeichnung (Stufenname) und Ressourcename. Achten Sie bei der Auswahl dieser Namen darauf, die CloudWatch Metrik-Grenzwerte nicht zu überschreiten.
- Die maximale Größe einer Zuordnungsvorlage beträgt 300 KB.

Amazon API Gateway wichtige Hinweise für WebSocket APIs

- API Gateway unterstützt Nachrichten-Payloads bis zu 128 KB mit einer maximalen Frame-Größe von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind. Wenn eine größere Nachricht empfangen wird, wird die Verbindung mit Code 1009 geschlossen.

Amazon API Gateway — wichtige Hinweise für REST APIs

- Das Pipe-Klartextzeichen (|) und das Zeichen für geschweifte Klammer ({ oder }) wird in Abfragezeichenfolgen zur Anforderung einer URL nicht unterstützt und muss URL-kodiert werden.
- Das Semikolon (;) wird in Abfragezeichenfolgen zur Anforderung einer URL nicht unterstützt und führt dazu, dass die Daten aufgeteilt werden.
- REST APIs dekodiert URL-kodierte Anforderungsparameter, bevor sie an Backend-Integrationen übergeben werden. Bei UTF-8-Anforderungsparametern APIs dekodiert REST die Parameter und übergibt sie dann als Unicode an Backend-Integrationen. APIs REST-URL-kodieren Sie das Prozentzeichen (%), bevor Sie es an Backend-Integrationen übergeben.
- Beim Verwenden der Amazon API Gateway-Konsole zum Testen einer API können Sie die Antwort "unknown endpoint errors" erhalten, wenn auf dem Backend ein selbstsigniertes Zertifikat präsentiert wird, in der Zertifikatkette ein Zwischenzertifikat fehlt oder das Backend eine andere, nicht erkennbare zertifikatbezogenen Ausnahme auslöst.
- Sie sollten eine API-[Resource](#) oder [Method](#)-Entität mit einer privaten Integration löschen, nachdem Sie hartcodierte Referenzen eines [VpcLink](#) entfernt haben. Andernfalls ist die Integration nicht abgeschlossen und Sie erhalten eine Fehlermeldung mit dem Hinweis, dass der VPC-Link noch verwendet wird, obwohl die Entität Resource oder Method gelöscht wurde. Dieses Verhalten liegt nicht vor, wenn die private Integration über eine Stufenvariable auf VpcLink verweist.
- Die folgenden Backends unterstützen die SSL-Clientauthentifizierung möglicherweise nicht auf eine mit API Gateway kompatible Weise:
 - [NGINX](#)
 - [Heroku](#)
- API Gateway unterstützt die meisten der [OpenAPI-2.0-Spezifikation](#) und der [OpenAPI-3.0-Spezifikation](#) mit den folgenden Ausnahmen:

- Pfadsegmente dürfen nur alphanumerische Zeichen, Unterstriche, Bindestriche, Punkte, Kommas, Doppelpunkte und geschweifte Klammern enthalten. Pfadparameter müssen als separate Pfadsegmente vorliegen. Beispiel: "resource/{path_parameter_name}" ist gültig, "resource{path_parameter_name}" nicht.
- Modellnamen dürfen nur alphanumerische Zeichen enthalten.
- Als Eingabeparameter werden nur die folgenden Attribute unterstützt: name, in, required, type, description. Andere Attribute werden ignoriert.
- Der securitySchemes-Typ muss bei Verwendung apiKey lauten. OAuth [2- und HTTP Basic-Authentifizierung werden jedoch über Lambda-Autorisierer unterstützt; die OpenAPI-Konfiguration wird über Herstellererweiterungen erreicht.](#)
- Das deprecated Feld wird nicht unterstützt und wird beim Export gelöscht. APIs
- API Gateway-Modelle werden unter Verwendung von [JSON-Schemaentwurf 4](#) anstelle des von OpenAPI verwendeten JSON-Schemas definiert.
- Der Parameter discriminator wird in Schemaobjekten nicht unterstützt.
- Das Tag example wird nicht unterstützt.
- Das Feld nullable wird nicht unterstützt.
- exclusiveMinimum wird von API Gateway nicht unterstützt.
- Die Tags maxItems und minItems werden bei der [einfachen Anforderungsvalidierung](#) nicht berücksichtigt. Um dieses Problem zu umgehen, aktualisieren Sie das Modell nach dem Import, bevor Sie die Validierung vornehmen.
- Für OpenAPI 3.0 dürfen oneOf, anyOf oder allOf nicht auf eine Definition innerhalb desselben Schemas über \$ref verweisen. Sie können Ihr Schema entweder direkt eingeben oder eine separate API-Gateway-Modellressource definieren. Weitere Informationen finden Sie unter [the section called "Erstellen komplexerer Modelle"](#).
- oneOf wird für OpenAPI 2.0 oder die SDK-Generierung nicht unterstützt.
- Das Feld readOnly wird nicht unterstützt.
- \$ref kann nicht für den Verweis auf andere Dateien verwendet werden.
- Antwortdefinitionen des Formulars "500": {"\$ref": "#/responses/UnexpectedError"} werden im OpenAPI-Basisverzeichnis nicht unterstützt. Ersetzen Sie die Referenz durch das Inline-Schema, um dieses Problem zu umgehen.
- Zahlen vom Typ Int32 oder Int64 werden nicht unterstützt. Ein Beispiel sehen Sie unten:

```
"elementId": {
```

```
"description": "Working Element Id",
"format": "int32",
"type": "number"
}
```

- Der Formattyp Dezimalzahl ("format": "decimal") wird in Schemadefinitionen nicht unterstützt.
- In Methodenantworten muss die Schemadefinition ein Objekttyp sein und darf keine primitiven Datentypen umfassen. Beispielsweise wird "schema": { "type": "string"} nicht unterstützt. Sie können dies jedoch umgehen, indem Sie den folgenden Objekttyp verwenden:

```
"schema": {
  "$ref": "#/definitions/StringResponse"
}

"definitions": {
  "StringResponse": {
    "type": "string"
  }
}
```

- API Gateway verwendet keine in der OpenAPI-Spezifikation definierte Sicherheit auf Stammebene. Daher muss die Sicherheit auf Vorgangsebene definiert werden, um korrekt angewendet werden zu können.
- Das default-Schlüsselwort wird nicht unterstützt.
- API Gateway führt die folgenden Beschränkungen und Limitierungen bei der Handhabung von Methoden mit Lambda- oder HTTP-Integration durch.
- Bei der Verarbeitung von Header-Namen und Abfrageparametern wird die Groß- und Kleinschreibung beachtet.
- Die folgende Tabelle listet die Header auf, die gelöscht, erneut zugewiesen oder anderweitig modifiziert werden können, wenn sie an den Integrationsendpunkt oder von diesem zurückgesendet werden. In dieser Tabelle:
 - Remapped bedeutet, dass der Header-Name von *<string>* in X-Amzn-Remapped-*<string>* geändert wird.

Remapped Overwritten bedeutet, dass der Header-Name von *<string>* in X-Amzn-Remapped-*<string>* geändert und der Wert überschrieben wird.

Header-Name	Anfrage (http/http_proxy /lambda)	Antwort (http/http_proxy /lambda)
Age	Pass-Through	Pass-Through
Accept	Pass-Through	Dropped/ Passthrough/ Passthrough
Accept-Charset	Pass-Through	Pass-Through
Accept-Encoding	Pass-Through	Pass-Through
Authorization	Pass-Through*	Remapped
Connection	Passthrough/Passthrough/Dropped	Remapped
Content-Encoding	Passthrough/Dropped/Passthrough	Pass-Through
Content-Length	Pass-Through (generiert auf der Grundlage des Inhalts)	Pass-Through
Content-MD5	Gelöscht	Remapped

Header-Name	Anfrage (http/http_proxy /lambda)	Antwort (http/http_proxy /lambda)
Content-Type	Pass-Through	Pass-Through
Date	Pass-Through	Neu zugeordnet überschrieben
Expect	Gelöscht	Gelöscht
Host	Auf den Integrationsendpunkt überschrieben	Gelöscht
Max-Forwards	Gelöscht	Remapped
Pragma	Pass-Through	Pass-Through
Proxy-Authenticate	Gelöscht	Gelöscht
Range	Pass-Through	Pass-Through
Referer	Pass-Through	Pass-Through

Header-Name	Anfrage (http/http_proxy /lambda)	Antwort (http/http_proxy /lambda)
Server	Gelöscht	Neu zugeordnet überschrieben
TE	Gelöscht	Gelöscht
Transfer-Encoding	Dropped/Dropped/Exception	Gelöscht
Trailer	Gelöscht	Gelöscht
Upgrade	Gelöscht	Gelöscht
User-Agent	Pass-Through	Remapped
Via	Dropped/Dropped/Passthrough	Passthrough/Dropped/Dropped
Warn	Pass-Through	Pass-Through
WWW-Authenticate	Gelöscht	Remapped

* Der Authorization-Header wird gelöscht, wenn er eine [Signaturversion 4](#)-Signatur enthält oder AWS_IAM-Autorisierung verwendet wird.

- Das Android-SDK einer von API Gateway generierten API verwendet die Klasse `java.net.HttpURLConnection`. Diese Klasse löst auf Geräten, auf denen Android 4.4 und

früher ausgeführt wird, eine unbehandelte Ausnahme für eine 401-Antwort aus, die aus der Neuordnung des Headers `WWW-Authenticate` zu `X-Amzn-Remapped-WWW-Authenticate` resultiert.

- Im Gegensatz zu API Gateway-generierten Java-, Android- und SDKs iOS-APIs unterstützt das JavaScript SDK einer von API Gateway generierten API keine Wiederholungsversuche bei Fehlern der Stufe 500.
- Der Test-Aufruf einer Methode verwendet den Standard-Inhaltstyp `application/json` und ignoriert Spezifikationen anderer Inhaltstypen.
- Beim Senden von Anfragen an eine API durch Übergabe des `X-HTTP-Method-Override`-Headers überschreibt API Gateway die Methode. Um den Header an das Backend zu übergeben, muss der Header der Integrationsanforderung hinzugefügt werden.
- Wenn eine Anfrage mehrere Medientypen in ihrem `Accept`-Header enthält, berücksichtigt API Gateway nur den ersten `Accept`-Medientyp. Wenn Sie die Reihenfolge der `Accept`-Medientypen nicht steuern können und der Medientyp Ihres binären Inhalts nicht der erste in der Liste ist, können Sie den ersten `Accept`-Medientyp in der Liste `binaryMediaTypes` Ihrer API hinzufügen. Amazon API Gateway gibt Ihren Inhalt dann im Binärformat zurück. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.
- Das Anpassen der Standard-Gateway-Antwort für 413 `REQUEST_TOO_LARGE` wird derzeit nicht unterstützt.
- API Gateway enthält einen `Content-Type`-Header für alle Integrationsantworten. Der Inhaltstyp ist standardmäßig „`application/json`“.

Dokumentverlauf

Die folgende Tabelle beschreibt die wichtigen Änderungen in der Dokumentation seit der letzten Version von Amazon API Gateway. Für Benachrichtigungen über Änderungen an dieser Dokumentation können Sie einen RSS-Feed abonnieren, indem Sie auf die RSS-Schaltfläche in der oberen Menüanzeige klicken.

- Letzte Aktualisierung der Dokumentation: 02. Dezember 2025

Änderung	Beschreibung	Datum
Fügen Sie eine API-Gateway-REST-API als Ziel für Amazon Bedrock AgentCore Gateway hinzu	API Gateway unterstützt jetzt das Hinzufügen einer REST-API als Ziel für Amazon Bedrock AgentCore Gateway. Weitere Informationen finden Sie unter Hinzufügen einer API-Gateway-REST-API als Ziel für Amazon Bedrock AgentCore Gateway .	2. Dezember 2025
Private Integration mit Application Load Balancer	API Gateway unterstützt jetzt private Integrationen mit Application Load Balancern für REST APIs. Weitere Informationen finden Sie unter Private Integrationen für REST APIs.	21. November 2025
Sicherheitsrichtlinien für REST APIs und benutzerdefinierte Domainnamen	API Gateway unterstützt jetzt erweiterte TLS-Sicherheitsrichtlinien für REST APIs und benutzerdefinierte Domainnamen. Weitere Informationen finden Sie unter Sicherheitsrichtlinien für REST APIs in API Gateway .	19. November 2025

[Antwort-Streaming für REST APIs](#)

API Gateway REST unterstützt APIs jetzt das progressive Streaming von Antwort-Payloads an Clients, sobald sie verfügbar sind. Weitere Informationen finden Sie unter [Streamen der Integrationsantwort für Ihre Proxy-Integrationen](#).

19. November 2025

[Entwicklerportale](#)

API Gateway unterstützt jetzt Portale, die als zentraler Ort dienen, an dem Ihre Kunden Ihre Produkte und Portalprodukte entdecken APIs und testen können, die einen Service oder eine Funktionalität darstellen, die Sie teilen möchten. Weitere Informationen finden Sie unter [API Gateway Gateway-Portale](#).

19. November 2025

[SIGv4a für REST APIs](#)

API Gateway unterstützt jetzt die Verwendung von Signature Version 4a für REST APIs. Weitere Informationen finden Sie unter [API-Gateway-Berechtigungsmodell für den Aufruf einer API](#).

19. August 2025

Routing-Regeln für REST APIs	API Gateway unterstützt jetzt Routing-Regeln für REST APIs unter Verwendung benutzerdefinierter Domainnamen. Eingehende Anfragen können dynamisch basierend auf HTTP-Header-Werten, URL-Basispfaden oder einer Kombination aus beidem weitergeleitet werden. Weitere Informationen finden Sie unter Senden von Traffic an Sie APIs über Ihren benutzerdefinierten Domainnamen .	3. Juni 2025
Unterstützung für Dual-Stack IPv4 - (und IPv6) Endpunkte hinzugefügt	API Gateway unterstützt jetzt Dual-Stack-Endpunkte für REST- WebSocket APIs, HTTP- und benutzerdefinierte Domainnamen. Weitere Informationen finden Sie unter IP-Adresstypen .	28. März 2025
Benutzerdefinierte Domainnamen für private APIs	Unterstützung für benutzerdefinierte Domainnamen für private Zwecke hinzugefügt APIs. Weitere Informationen finden Sie unter Benutzerdefinierte Domainnamen für private Zwecke APIs .	21. November 2024

[Zusätzlicher Support für TLS 1.3](#)

API Gateway unterstützt jetzt TLS 1.3 auf regionalen REST-APIs, HTTP- und WebSocket APIs. Weitere Informationen finden Sie unter [Auswählen einer Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway](#).

15. Februar 2024

[Aktualisierungen der WebSocket REST-API und der API-Konsole](#)

Aktualisierte Konsoleninformationen für REST-APIs und WebSocket APIs.

10. Dezember 2023

[Aktualisierung der Dokumentation](#)

Die konzeptionellen Informationen wurden aktualisiert und neue Tutorials für Datentransformationen und Themen zur Anforderungvalidierung für API Gateway REST-APIs erstellt. Weitere Informationen finden Sie unter [Die Anforderungvalidierung in API Gateway verwenden](#) und [Datentransformationen für die REST-API einrichten](#).

22. Juni 2023

[DNS-Failover für ein API Gateway mit mehreren Regionen konfigurieren](#)

Unterstützung für die Verwendung von Amazon Route 53-Zustandsprüfungen zur Steuerung des DNS-Failovers von einer API-Gateway-REST-API in einer primären AWS-Region zu einer in einer sekundären Region hinzugefügt. Weitere Informationen finden Sie unter [Benutzerdefinierte Integritätsprüfungen für DNS-Failover konfigurieren](#).

31. Oktober 2022

[Aktualisierung der Dokumentation](#)

Die Zusammenfassungen der wichtigsten Funktionen für die REST-API und die HTTP-API wurden aktualisiert. APIs Weitere Informationen finden Sie unter [Auswahl zwischen REST-API und HTTP-API APIs](#).

31. Mai 2022

[Aktualisierung der verwalteten Richtlinien](#)

acm:GetCertificate - Unterstützung für AWSServiceRoleForAPIGateway - Richtlinien wurde hinzugefügt. Weitere Informationen finden Sie unter [Verwenden von serviceverknüpften Rollen für API Gateway](#).

12. Juli 2021

[Parameterzuordnung für HTTP APIs](#)

Unterstützung für die Parameterzuordnung für HTTP hinzugefügt APIs. Weitere Informationen finden Sie unter [Transformieren von API-Anfragen und -Antworten](#).

7. Januar 2021

[Standard-Endpunkt für eine REST-API deaktivieren](#)

Unterstützung für die Deaktivierung des Standardendpunkts für REST APIs hinzugefügt. Weitere Informationen finden Sie unter [Standard-Endpunkts für eine REST-API deaktivieren](#).

29. Oktober 2020

[Gegenseitige TLS-Authentifizierung](#)

Unterstützung für die gegenseitige TLS-Authentifizierung für REST APIs und HTTP APIs hinzugefügt. Weitere Informationen finden Sie unter [Konfigurieren der gegenseitigen TLS-Authentifizierung für eine REST-API](#) und [Konfigurieren der gegenseitigen TLS-Authentifizierung für eine HTTP-API](#).

17. September 2020

[AWS Lambda HTTP-API-Autorisierer](#)

Unterstützung für AWS Lambda Autorisierer für HTTP hinzugefügt. APIs Weitere Informationen finden Sie unter [Arbeiten mit AWS Lambda Autorisierern](#) für HTTP. APIs

9. September 2020

[HTTP-API-Dienstintegrationen AWS](#)

Unterstützung für AWS Serviceintegrationen für HTTP hinzugefügt. APIs Weitere Informationen finden Sie unter [Arbeiten mit AWS Dienstintegrationen für HTTP](#). APIs

20. August 2020

[Benutzerdefinierte HTTP-API- Platzhalterdomänen](#)

Unterstützung für benutzerdefinierte Domainnamen mit Platzhaltern für HTTP hinzugefügt. APIs Weitere Informationen finden Sie unter [Benutzerdefinierte Domännennamen mit Platzhalter](#).

10. August 2020

[Verbesserungen des serverlosen Entwicklerportals](#)

Benutzerverwaltung zum Administratorbereich und Support für den Export von API-Definitionen hinzugefügt. Weitere Informationen finden Sie unter [Verwenden Sie das serverlose Entwicklerportal, um Ihr API Gateway APIs zu katalogisieren](#).

25. Juni 2020

[WebSocket API-Unterstützung Sec-WebSocket-Protocol](#)

Unterstützung für das Sec-WebSocket-Protocol - Feld hinzugefügt. Weitere Informationen finden Sie unter [Eine \\$connect-Route einrichten, für die ein WebSocket Unterprotokoll erforderlich ist](#).

16. Juni 2020

HTTP-API-Export	Unterstützung für den Export von OpenAPI 3.0-Definitionen von HTTP APIs hinzugefügt. Weitere Informationen finden Sie unter HTTP-API von API Gateway exportieren .	20. April 2020
Sicherheitsdokumentation	Sicherheitsdokumentation hinzugefügt. Weitere Informationen finden Sie unter Sicherheit in Amazon API Gateway .	31. März 2020
Neu strukturierte Dokumentation	Das Entwicklerhandbuch wurde neu strukturiert.	12. März 2020
Allgemeine Verfügbarkeit der HTTP-API	HTTP ist jetzt APIs allgemein verfügbar. Weitere Informationen finden Sie unter Arbeiten mit HTTP APIs .	12. März 2020
HTTP-API-Protokollierung	Unterstützung für <code>\$context.integrationErrorMessage</code> in HTTP-API-Protokollen hinzugefügt. Weitere Informationen finden Sie unter HTTP-API-Protokollierungsvariablen .	26. Februar 2020
AWS Variablen für den OpenAPI-Import	Unterstützung für AWS Variablen in OpenAPI-Definitionen hinzugefügt. Weitere Informationen finden Sie unter AWS -Variablen für OpenAPI-Import .	17. Februar 2020

HTTP APIs	HTTP wurde APIs in der Betaversion veröffentlicht. Weitere Informationen finden Sie unter HTTP APIs .	4. Dezember 2019
Benutzerdefinierte Domänennamen mit Platzhalter	Unterstützung für benutzerdefinierte Domänennamen mit Platzhalter hinzugefügt. Weitere Informationen finden Sie unter Benutzerdefinierte Domänennamen mit Platzhalter .	21. Oktober 2019
Amazon-Data-Firehose-Protokollierung	Zusätzlicher Support für Amazon Data Firehose als Ziel für Zugriffsprotokolldaten. Weitere Informationen finden Sie unter Verwenden von Amazon Data Firehose als Ziel für die API-Gateway-Zugriffsprotokollierung .	15. Oktober 2019
Route53-Aliase zum Aufrufen von Private APIs	Unterstützung für zusätzliche Route53-Alias-DNS-Einträge zum Aufrufen von Private hinzugefügt. APIs Weitere Informationen finden Sie unter Aufrufen Ihrer privaten API mit Route53-Alias .	18. September 2019
Tag-basierte Zugriffskontrolle für WebSocket APIs	Unterstützung für die Tag-basierte Zugriffskontrolle für hinzugefügt. WebSocket APIs Weitere Informationen finden Sie unter API Gateway-Ressourcen, die markiert werden können .	27. Juni 2019

[Auswahl der TLS-Version für benutzerdefinierte Domänen](#)

Unterstützung für die Auswahl von TLS-Versionen (Transport Layer Security) für Versionen hinzugefügt APIs, die in benutzerdefinierten Domänen bereitgestellt werden. Siehe Hinweis in [TLS-Mindestversion für eine benutzerdefinierte Domäne in API Gateway auswählen](#).

20. Juni 2019

[VPC-Endpunktrichtlinien für private APIs](#)

Unterstützung für die Verbesserung der Sicherheit von Private APIs durch das Anhängen von Endpunktrichtlinien an VPC-Schnittstellen-Endpunkte wurde hinzugefügt. Weitere Informationen finden Sie unter [Verwenden von VPC-Endpunktrichtlinien für Private APIs in API Gateway](#).

4. Juni 2019

[Dokumentation aktualisiert](#)

[Einstieg mit Amazon API Gateway](#) wurde neu erstellt. Tutorials nach [Amazon API Gateway Tutorials](#) verschoben.

29. Mai 2019

[Tag-basierte Zugriffskontrolle für REST APIs](#)

Unterstützung für Tag-basierte Zugriffskontrolle für REST hinzugefügt. APIs Weitere Informationen finden Sie unter [Tags mit IAM-Richtlinien zur Steuerung des Zugriffs auf API Gateway-Ressourcen verwenden](#).

23. Mai 2019

Dokumentation aktualisiert

Es wurden sechs Themen überarbeitet: [Was ist Amazon API Gateway?](#), [Tutorial: Erstellen einer API mit HTTP-Proxy-Integration](#), [Tutorial: Erstellen einer Calc-REST-API mit drei Nicht-Proxy-Integrationen](#), [Referenz für die API Gateway-Zuweisungsvorlage und Zugriffsprotokollierungsvariable](#), [Verwenden von API Gateway Lambda-Genehmigern](#) und [Aktivieren von CORS für eine API Gateway-REST-API-Ressource](#).

5. April 2019

Verbesserungen des serverlosen Entwicklerportals

Administrator-Panel und weitere Verbesserungen wurden hinzugefügt, um die Veröffentlichung APIs im Amazon API Gateway Gateway-Entwicklerportal zu vereinfachen. Weitere Informationen finden Sie unter [Verwenden Sie ein Entwicklerportal, um Ihre zu katalogisieren APIs](#).

28. März 2019

Support für AWS Config

Unterstützung für hinzugefügt AWS Config. Weitere Informationen finden Sie unter [Überwachung der API-Gateway-API-Konfiguration mit AWS Config](#).

20. März 2019

Support für CloudFormation	API Gateway V2 API zur CloudFormation Vorlagenreferenz hinzugefügt. Weitere Informationen finden Sie unter Amazon API Gateway V2-Ressourcentypenreferenz .	7. Februar 2019
Support für WebSocket APIs	Unterstützung für hinzugefügt WebSocket APIs. Weitere Informationen finden Sie unter Erstellen einer WebSocket API in Amazon API Gateway .	18. Dezember 2018
Serverloses Entwicklerportal verfügbar über AWS Serverless Application Repository	Die serverlose Anwendung für das Amazon API Gateway Gateway-Entwicklerportal ist jetzt unter AWS Serverless Application Repository (zusätzlich zu GitHub) verfügbar. Weitere Informationen finden Sie unter Verwenden Sie ein Entwicklerportal, um Ihr API Gateway zu katalogisieren APIs .	16. November 2018
Support für AWS WAF	Zusätzliche Unterstützung für AWS WAF (Web Application Firewall). Weitere Informationen finden Sie unter Steuern des Zugriffs auf eine API mithilfe von AWS WAF .	5. November 2018

[Serverloses Entwicklerportal](#)

Amazon API Gateway bietet jetzt ein vollständig anpassbares Entwicklerportal als serverlose Anwendung, das Sie für die Veröffentlichung Ihres API Gateway APIs einsetzen können. Weitere Informationen finden Sie unter [Verwenden Sie ein Entwicklerportal, um Ihr API Gateway zu katalogisieren APIs.](#)

29. Oktober 2018

[Unterstützung für mehrwertige Header- und Abfragezeichenfolgenparameter](#)

Amazon API Gateway unterstützt jetzt mehrere gleichnamige Header und Abfragezeichenfolgenparameter. Weitere Informationen finden Sie unter [Unterstützung für mehrwertige Header- und Abfragezeichenfolgenparameter.](#)

4. Oktober 2018

[OpenAPI-Support](#)

Amazon API Gateway unterstützt jetzt sowohl OpenAPI 3.0 als auch OpenAPI (Swagger) 2.0.

27. September 2018

[Dokumentation aktualisiert](#)

Ein neues Thema wurde hinzugefügt: [So wirken sich Ressourcenrichtlinien von Amazon API Gateway auf den Autorisierungs-Workflow aus.](#)

27. September 2018

Aktive AWS X-Ray Integration

Sie können es jetzt verwenden AWS X-Ray , um Latenzen bei Benutzeranfragen zu verfolgen und zu analysieren, während diese von Ihnen APIs zu den zugrundeliegenden Diensten weitergeleitet werden. Weitere Informationen finden Sie unter [Trace API Gateway API Execution with AWS X-Ray](#).

6. September 2018

Caching-Verbesserungen

Nur GET-Methoden haben standardmäßig Caching aktiviert, wenn Sie das Caching für eine API-Phase aktivieren. Dies hilft, die Sicherheit Ihrer API zu gewährleisten. Sie können das Caching für andere Methoden aktivieren, indem Sie die Methodeneinstellungen überschreiben. Weitere Informationen finden Sie unter [Aktivieren von API-Caching für verbesserte Reaktionsfähigkeit](#).

20. August 2018

Service Limits überarbeitet

Es wurden mehrere Grenzwerte überarbeitet: Die Anzahl der APIs pro Konto wurde erhöht. Erhöhte API-Ratenlimits für Create/Import/Deploy APIs. Einige Raten von pro Minute auf pro Sekunde korrigiert. Weitere Informationen finden Sie unter [Limits](#).

13. Juli 2018

[Überschreiben von API-Anforderungs- und Antwortparametern und Headern](#)

Unterstützung für Überschreibung von Anforderungsheader n, Abfragestrings und Pfaden sowie für Antwort-Header und Statuscodes hinzugefügt. Weitere Informationen finden Sie unter [Verwendung einer Zuweisungsvorlage zum Überschreiben der Anfrage- und Antwortparameter und -Header einer API](#).

12. Juli 2018

[Drosselung auf Methodenebene für Nutzungspläne](#)

Unterstützung für die Einrichtung von Standard-Drossel-Limits pro Methode sowie zur Einstellung von Drossel-Limits für individuelle API-Methoden in den Nutzungsplan-Einstellungen hinzugefügt. Diese Einstellungen bestehen zusätzlich zu den bestehenden Drossel-Limits auf Kontoebene und Standard-Drossel-Limits auf Methodenebene, die Sie in den Stufeneinstellungen festlegen können. Weitere Informationen finden Sie unter [Drosselungs-API-Anfragen für einen besseren Durchsatz](#).

11. Juli 2018

[API Gateway Entwicklerhandbuch Update-Benachrichtigungen jetzt über RSS verfügbar](#)

Die HTML-Version des API Gateway Entwicklerhandbuchs unterstützt jetzt einen RSS-Feed mit Aktualisierungen, die auf dieser Dokumentenhistorie-Seite dokumentiert sind. Der RSS-Feed umfasst Aktualisierungen ab 27. Juni 2018 und später. Zuvor angekündigte Updates stehen weiterhin auf dieser Seite zur Verfügung. Verwenden Sie die RSS-Schaltfläche in der oberen Menüanzeige, um den Feed zu abonnieren.

27. Juni 2018

Frühere Updates

Die folgende Tabelle beschreibt wichtige Änderungen in jeder Version des API Gateway Entwicklerhandbuchs vor dem 27. Juni 2018.

Änderungen	Beschreibung	Datum geändert
Privat APIs	Es wurde Unterstützung für private Geräte hinzugefügt APIs, die Sie über VPC-Endpunkte der Schnittstelle verfügbar machen. Der Datenverkehr zu Ihrem privaten APIs Netzwerk verlässt das Amazon-Netzwerk nicht; er ist vom öffentlichen Internet isoliert.	14. Juni 2018
Kontoübergreifende Lambda-Autorisierungen und -Integrationen und kontoübergreifende Amazon Cognito-B	Verwenden Sie eine AWS Lambda Funktion aus einem anderen AWS Konto als Lambda-Autorisierungsfunktion oder als API-Integrations-Backend. Oder verwenden Sie einen Amazon Cognito-Benutzerpool als Autorisierungsfunktion. Das andere Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist.	2. April 2018

Änderungen	Beschreibung	Datum geändert
Benutzerpool-Autorisierungen	Weitere Informationen finden Sie unter the section called “Kontenübergreifenden Lambda-Genehmiger konfigurieren” , the section called “Tutorial: REST-API mit kontenübergreifender Lambda-Proxy-Integration erstellen” und the section called “Kontenübergreifenden Amazon Cognito-Genehmiger für eine REST-API konfigurieren” .	
Ressourcenrichtlinien für APIs	Mit Amazon-API-Gateway-Ressourcenrichtlinien ermöglichen Sie Benutzern eines anderen AWS -Kontos, sicher auf Ihre API zuzugreifen oder legen fest, dass die API nur von einem bestimmten IP-Adressbereich oder bestimmten CIDR-Blöcken aufgerufen werden kann. Weitere Informationen finden Sie unter the section called “API Gateway-Ressourcenrichtlinien verwenden” .	2. April 2018
Markieren bei API Gateway-Ressourcen	Markieren Sie eine API-Stufe mit bis zu 50 Tags für die Kostenzuweisung von API-Anfragen und das Caching in API Gateway. Weitere Informationen finden Sie unter the section called “Einrichten von Tags” .	19. Dezember 2017
Nutzlastkomprimierung und -dekomprimierung	Aktivieren Sie den Aufruf Ihrer API mit komprimierten Nutzlasten mithilfe einer der unterstützten Inhaltskodierungen. Die komprimierten Nutzlasten werden zugeordnet, wenn eine Textzuweisungsvorlage angegeben wird. Weitere Informationen finden Sie unter the section called “Inhaltskodierung” .	19. Dezember 2017
Von einem Genehmiger stammender API-Schlüssel	Geben Sie einen API-Schlüssel von einem benutzerdefinierten Genehmiger an API Gateway zurück, um einen Nutzungsplan für API-Methoden anzuwenden, die den Schlüssel erfordern. Weitere Informationen finden Sie unter the section called “Auswählen einer API-Schlüsselquelle in API Gateway” .	19. Dezember 2017

Änderungen	Beschreibung	Datum geändert
Autorisierung mit OAuth 2 Bereichen	Aktivieren Sie die Autorisierung des Methodenaufrufs, indem Sie OAuth 2 Bereiche mit dem Autorisierer verwenden. COGNITO_USER_POOLS Weitere Informationen finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden” .	14. Dezember 2017
Private Integration und VPC-Link	Erstellen Sie eine API mit der privaten API Gateway Gateway-Integration, um Kunden den Zugriff auf HTTP/HTTPS Ressourcen in einer Amazon-VPC von außerhalb der VPC über eine VpcLink Ressource zu ermöglichen. Weitere Informationen erhalten Sie unter the section called “Tutorial: Erstellen einer REST-API mit einer privaten Integration” und the section called “Richten Sie eine private Integration ein” .	30. November 2017
Bereitstellen eines Canary für API-Tests	Fügen Sie einer vorhandenen API-Bereitstellung ein Canary-Release hinzu, um eine neuere Version der API zu testen, während die aktuelle Version auf derselben Stufe in Betrieb ist. Sie können einen Prozentsatz des Stage-Traffics für die Canary-Version festlegen und die Ausführung und den Zugriff auf bestimmte Kanaren aktivieren, die in separaten Log-Logs protokolliert werden. CloudWatch Weitere Informationen finden Sie unter the section called “Einrichten einer Canary-Release-Bereitstellung” .	28. November 2017
Zugriffsprotokollierung	Protokollieren Sie den Client-Zugriff auf Ihre API mit Daten aus \$context-Variablen in einem Format Ihrer Wahl. Weitere Informationen finden Sie unter the section called “CloudWatch protokolliert” .	21. November 2017

Änderungen	Beschreibung	Datum geändert
Ruby-SDK auf einer API	Erstellen Sie ein Ruby-SDK für Ihre API und nutzen Sie diesen für den Aufruf Ihrer API-Methoden. Weitere Informationen erhalten Sie unter the section called “Generieren des Ruby SDK einer API in API Gateway” und the section called “Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden” .	20. November 2017
Regionaler API-Endpunkt	Geben Sie einen regionalen API-Endpunkt an, um eine API für nicht mobile Clients zu erstellen. Ein Client, der nicht mobil ist, z. B. eine EC2 Instance, wird in derselben AWS Region ausgeführt, in der die API bereitgestellt wird. Wie bei einer Edge-optimierten API können Sie einen benutzerdefinierten Domännennamen für eine regionale API erstellen. Weitere Informationen erhalten Sie unter the section called “API-Gateway-Endpunkttypen” und the section called “Einrichten eines regionalen benutzerdefinierten Domainnamens” .	2. November 2017
Benutzerdefinierte Genehmiger einer Anforderung	Verwenden Sie den benutzerdefinierten Genehmiger für eine Anforderung, um Benutzer-Authentifizierungsinformationen in Anforderungsparametern bereitzustellen und API-Methodenaufrufe zu genehmigen. Zu den Anforderungsparametern gehören Header- und Abfragezeichenfolgenparameter, ebenso wie Stufen- und Kontextvariablen. Weitere Informationen finden Sie unter API Gateway-Lambda-Genehmiger verwenden .	15. September 2017

Änderungen	Beschreibung	Datum geändert
Anpassen von Gateway-Antworten	Anpassen von API Gateway-generierten Gateway-Antworten auf API-Anfragen, die das Integrations-Backend nicht erreichen. Eine angepasste Gateway-Nachricht kann API-spezifische benutzerdefinierte Fehlermeldungen für den Aufrufer enthalten – darunter die erforderlichen CORS-Header – oder die Gateway-Antwortdaten in ein externes Austauschformat umwandeln. Weitere Informationen finden Sie unter Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen .	6. Juni 2017
Benutzerdefinierte Lambda-Fehlereigenschaften auf Methoden-Antwortheader zuordnen	Ordnen Sie einzelne benutzerdefinierte Fehlereigenschaften, die von Lambda zurückgegeben werden, den Antwort-Header-Parametern der Methode mit dem <code>integration.response.body</code> -Parameter zu, wobei API Gateway zur Deserialisierung des benutzerdefinierten Zeichenfolgen-Fehlerobjekts zur Laufzeit verwendet wird. Weitere Informationen finden Sie unter Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln .	6. Juni 2017
Erhöhen der Drosselungslimits	Erhöhen Sie die Begrenzung der konstanten Anforderungsrates auf Kontoebene auf 10 000 Anforderungen pro Sekunde (RPS) und die Burst-Begrenzung auf 5 000 gleichzeitige Anforderungen. Weitere Informationen finden Sie unter Drosseln Sie Anfragen an Ihren REST APIs für einen besseren Durchsatz in API Gateway .	6. Juni 2017
Validieren von Methodenanforderungen	Konfigurieren Sie grundlegende Anfragevalidierer auf API- oder Methodenebene, sodass API Gateway eingehende Anfragen validieren kann. Amazon API Gateway überprüft, ob die erforderlichen Parameter festgelegt und nicht leer sind und ob das Format der anwendbaren Nutzlasten mit dem konfigurierten Modell konform ist. Weitere Informationen finden Sie unter Validierung für REST APIs in API Gateway anfordern .	11. April 2017

Änderungen	Beschreibung	Datum geändert
Integration in ACM	Verwenden Sie ACM-Zertifikate für die benutzerdefinierte n Domänennamen Ihrer API. Sie können ein Zertifikat in ACM erstellen AWS Certificate Manager oder ein vorhandenes Zertifikat im PEM-Format in ACM importieren. Sie beziehen sich dann auf den ARN des Zertifikats, wenn Sie einen benutzerdefinierten Domainnamen für Ihren festlegen APIs. Weitere Informationen finden Sie unter Benutzerdefinierter Domainname für öffentliches REST APIs in API Gateway .	9. März 2017
Generieren und Aufrufen des Java-SDKs einer API	Lassen Sie API Gateway das Java-SDK für Ihre API generieren und verwenden Sie das SDK, um die API in Ihrem Java-Client aufzurufen. Weitere Informationen finden Sie unter Von API Gateway generiertes Java-SDK für eine REST-API verwenden .	13. Januar 2017
Integration mit AWS Marketplace	Verkaufen Sie Ihre API in einem Nutzungsplan als SaaS-Produkt über AWS Marketplace. und erhöhen Sie mit AWS Marketplace die Reichweite für Ihre API. Verlassen Sie sich AWS Marketplace auf die Kundenabrechnung in Ihrem Namen. Überlassen Sie API Gateway die Benutzerautorisierung und Nutzungsmessung. Weitere Informationen finden Sie unter Verkaufen Sie Ihre APIs als SaaS .	1. Dezember 2016
Aktivieren der Dokumentationsunterstützung für Ihre API	Fügen Sie Dokumentation für API-Entitäten in DocumentationPart Ressourcen in API Gateway hinzu. Ordnen Sie einen Snapshot der DocumentationPart Sammlungen instanzen einer API-Phase zu, um eine zu erstellen DocumentationVersion . Veröffentlichen Sie die API-Dokumentation durch Exportieren einer Dokumentationsversion in eine externe Datei, beispielsweise eine OpenAPI-Datei. Weitere Informationen finden Sie unter Dokumentation für REST APIs in API Gateway .	1. Dezember 2016

Änderungen	Beschreibung	Datum geändert
Aktualisierung des benutzerdefinierten Genehmigers	Eine Kunden-Genehmiger-Lambda-Funktion gibt jetzt die Prinzipal-ID des Aufrufers zurück. Die Funktion kann auch andere Informationen zurückgeben, darunter Schlüssel-Wert-Paare der context-Zuordnung und eine IAM-Richtlinie. Weitere Informationen finden Sie unter Ausgabe von einem API-Gateway-Lambda-Genehmiger .	1. Dezember 2016
Unterstützung binärer Nutzlasten	Stellen binaryMediaTypes Sie Ihre API so ein, dass binäre Payloads einer Anfrage oder Antwort unterstützt werden. Legen Sie die <code>contentHandling</code> Eigenschaft für eine Integration fest oder geben Sie IntegrationResponse an, ob eine binäre Nutzlast als systemeigener Binär-Blob, als Base64-kodierte Zeichenfolge oder als Passthrough ohne Änderungen behandelt werden soll. Weitere Informationen finden Sie unter Binäre Medientypen für REST APIs in API Gateway .	17. November 2016
Aktivieren einer Proxy-Integration mit einem HTTP- oder Lambda-Backend über die Proxy-Ressource einer API.	Erstellen Sie eine Proxy-Ressource mit einem Greedy-Path-Parameter der Form <code>{proxy+}</code> und der Catch-all ANY-Methode. Die Proxy-Ressource wird mit Hilfe der HTTP- bzw. Lambda-Proxy-Integration in ein HTTP- oder Lambda-Backend integriert. Weitere Informationen finden Sie unter Einrichten der Proxy-Integration mit einer Proxy-Ressource .	20. September 2016
Erweiterung APIs der in API Gateway ausgewählten Produktangebote für Ihre Kunden durch die Bereitstellung eines oder mehrerer Nutzungspläne.	Erstellen Sie einen Nutzungsplan in API Gateway, um ausgewählten API-Kunden den Zugriff auf bestimmte API-Stufen zu vereinbarten Anfrageraten und Quoten zu ermöglichen. Weitere Informationen finden Sie unter Nutzungspläne und API-Schlüssel für REST APIs in API Gateway .	11. August 2016

Änderungen	Beschreibung	Datum geändert
Autorisierung auf Methodenebene mit einem Benutzerpool in Amazon Cognito ermöglichen	Erstellen Sie einen Benutzerpool in Amazon Cognito und verwenden Sie ihn als Ihren eigenen Identitätsanbieter . Sie können den Benutzerpool für die Autorisierung auf Methodenebene konfigurieren, um Benutzern, die im Benutzerpool registriert sind, Zugriff zu gewähren. Weitere Informationen finden Sie unter Steuern Sie den Zugriff auf REST APIs mithilfe von Amazon Cognito Benutzerpools als Autorisierer .	28. Juli 2016
Aktivierung von CloudWatch Amazon-Metriken und -Dimensionen unter dem AWS/ApiGateway Namespace.	Die API Gateway Gateway-Metriken sind jetzt unter dem CloudWatch Namespace von AWS/ApiGateway standardisiert. Sie können sie sowohl in der API Gateway Gateway-Konsole als auch in der CloudWatch Amazon-Konsole anzeigen. Weitere Informationen finden Sie unter Amazon API Gateway-Dimensionen und -Metriken .	28. Juli 2016
Aktivieren der Zertifikatrotation für einen benutzerdefinierten Domännennamen	Mit der Zertifikatrotation können Sie ein ablaufendes Zertifikat für einen benutzerdefinierten Domännennamen hochladen und erneuern. Weitere Informationen finden Sie unter Wechseln eines in ACM importierten Zertifikats .	27. April 2016
Dokumentationsänderungen für die aktualisierte Amazon API Gateway-Konsole	Erfahren Sie, wie Sie eine API mit der aktualisierten API Gateway-Konsole erstellen und einrichten. Weitere Informationen erhalten Sie unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels und Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen .	5. April 2016

Änderungen	Beschreibung	Datum geändert
Aktivieren der Funktion zum Importieren von APIs, um aus externen API-Definitionen neue APIs zu erstellen oder vorhandene zu aktualisieren	Mit der Funktion zum Importieren von APIs erstellen Sie eine neue API oder aktualisieren eine vorhandene, indem Sie mit den API Gateway-Erweiterungen eine externe, in Swagger 2.0 angegebene API-Definition hochladen. Weitere Informationen über die Funktion zum Importieren von APIs erhalten Sie unter REST-APIs mit OpenAPI in API Gateway erstellen .	5. April 2016
Offenlegen der <code>\$input.body</code> - Variablen, um auf die rohe Nutzlast als Zeichenfolge zuzugreifen, und der <code>\$util.parseJson()</code> - Funktion, um eine JSON-Zeichenfolge in ein JSON-Objekt in einer Zuweisungsvorlage zu verwandeln	Weitere Informationen zu <code>\$input.body</code> und <code>\$util.parseJson()</code> finden Sie unter Variablen für Datentransformationen für API Gateway .	5. April 2016
Aktivieren von Client-Anforderungen mit Cache-Invalidierung auf Methodenebene und Verbesserung der Verwaltung der Anforderungsdrosselung	Leeren Sie den Cache auf der Ebene der API-Stufen oder heben Sie die Gültigkeit einzelner Cache-Einträge auf. Weitere Informationen erhalten Sie unter API-Stufen-Cache in API Gateway leeren und API Gateway-Cache-Eintrag ungültig machen . Verbessern Sie die Konsolenumgebung für die Verwaltung der Drosselung von API-Anforderungen. Weitere Informationen finden Sie unter Drosseln Sie Anfragen an Ihren REST APIs für einen besseren Durchsatz in API Gateway .	25. März 2016

Änderungen	Beschreibung	Datum geändert
API Gateway-API mithilfe von benutzerdefinierte r Autorisierung aktivieren und aufrufen	Erstellen und konfigurieren Sie eine AWS Lambda Funktion zur Implementierung einer benutzerdefinierten Autorisierung. Die Funktion gibt ein IAM-Richtliniendokument zurück, das die Allow- oder Deny-Berechtigungen für Client-Anfragen einer API Gateway-API gewährt. Weitere Informationen finden Sie unter API Gateway-Lambda-Generierungsmuster verwenden .	11. Februar 2016
API Gateway-API unter Verwendung einer Swagger-Definitionsdatei und Erweiterungen importieren und exportieren	Erstellen und aktualisieren Sie Ihr API Gateway mithilfe der Swagger-Spezifikation mit den API Gateway-Erweiterungen. Importieren Sie die OpenAPI-Definitionen mit der API Gateway-Importfunktion. Exportieren Sie eine API Gateway-API in eine Swagger-Definitionsdatei unter Verwendung der API Gateway-Konsole oder der API Gateway Export-API. Weitere Informationen erhalten Sie unter REST-APIs mit OpenAPI in API Gateway erstellen und REST-API von API Gateway importieren .	18. Dezember 2015
Zuordnung von Anforderungs- oder Antworttext bzw. JSON-Feldern im Text zu Anforderungs- oder Antwortparametern	Weisen Sie den Methodenanforderungstext oder dessen JSON-Felder dem Pfad, der Abfragezeichenfolge oder den Headern der Integrationsanforderung zu. Ordnen Sie den Integrationsantworttext oder dessen JSON-Felder den Antwort-Headern der Anforderung zu. Weitere Informationen finden Sie unter Beispiele für Parameterzuweisungen für REST-APIs in API Gateway .	18. Dezember 2015
Mit Stage-Variablen in Amazon API Gateway arbeiten	Erfahren Sie, wie Konfigurationsattribute mit einer Bereitstellungsphase einer API in Amazon API Gateway verknüpft werden können. Weitere Informationen finden Sie unter Stufenvariablen für eine REST-API in API Gateway verwenden .	5. November 2015

Änderungen	Beschreibung	Datum geändert
Vorgehensweise: Aktivieren von CORS für eine Methode	Es ist jetzt einfacher, Cross-Origin Resource Sharing (CORS) für Methoden in Amazon API Gateway zu aktivieren. Weitere Informationen finden Sie unter CORS für REST APIs im API Gateway .	3. November 2015
Vorgehensweise: Verwenden von clientseitiger SSL-Authentifizierung	Verwenden Sie Amazon API Gateway zur Generierung von SSL-Zertifikaten, die Sie zur Authentifizierung von Aufrufen an Ihr HTTP-Backend verwenden können. Weitere Informationen finden Sie unter Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-Authentifizierung in API Gateway .	22. September 2015
Pseudointegration von Methoden	Erfahren Sie, wie die Mock-Integration einer API mit Amazon API Gateway funktioniert. Diese Funktion ermöglicht es Entwicklern, API-Antworten direkt über API Gateway zu generieren, ohne dass zuvor ein endgültiges Integrations-Backend erforderlich ist.	1. September 2015
Unterstützung von Amazon Cognito Identity	Amazon API Gateway hat den Bereich der <code>\$context</code> -Variablen erweitert, sodass jetzt Informationen über Amazon Cognito Identity zurückgegeben werden, wenn Anfragen mit Amazon Cognito-Anmeldeinformationen signiert werden. Darüber hinaus haben wir eine <code>\$util</code> Variable hinzugefügt, mit der Zeichen in Zeichenketten maskiert JavaScript URLs und kodiert werden können. Weitere Informationen finden Sie unter Variablen für Datentransformationen für API Gateway .	28. August 2015

Änderungen	Beschreibung	Datum geändert
OpenAPI-Integration	Verwenden Sie das Swagger-Importtool auf GitHub , um Swagger-API-Definitionen in Amazon API Gateway zu importieren. Erfahren Sie mehr über OpenAPI-Erweiterungen für API Gateway das Erstellen und Bereitstellen APIs und die Methoden mit dem Importtool. Mit dem Swagger-Importer-Tool können Sie auch bestehende aktualisieren. APIs	21. Juli 2015
Referenz für die Zuweisungsvorlage	Erfahren Sie unter Variablen für Datentransformationen für API Gateway mehr über den <code>\$input</code> -Parameter und seine Funktionen.	18. Juli 2015
Erste veröffentlichte Version	Dies ist die erste öffentliche Version des API Gateway-Entwicklerhandbuch.	9. Juli 2015

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.