



Panduan Developer

Amazon API Gateway



Amazon API Gateway: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon adalah milik dari pemiliknya masing-masing, yang mungkin berafiliasi atau tidak berafiliasi dengan, terkait, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon API Gateway?	1
Arsitektur API Gateway	2
Fitur API Gateway	2
Kasus penggunaan API Gateway	3
Gunakan API Gateway untuk membuat REST APIs	4
Gunakan API Gateway untuk membuat HTTP APIs	5
Gunakan API Gateway untuk membuat WebSocket APIs	5
Siapa yang menggunakan API Gateway?	6
Mengakses API Gateway	7
Bagian dari infrastruktur AWS tanpa server	7
Cara memulai dengan Amazon API Gateway	8
Konsep API Gateway	8
Pilih antara REST APIs dan HTTP APIs	14
Tipe titik akhir	14
Keamanan	14
Otorisasi	15
Manajemen API	15
Pengembangan	16
Memantau	17
Integrasi	17
Memulai dengan konsol REST API	18
Langkah 1: Membuat fungsi Lambda	19
Langkah 2: Buat REST API	20
Langkah 3: Buat integrasi proxy Lambda	20
Langkah 4: Menerapkan API Anda	20
Langkah 5: Panggil API Anda	21
(Opsional) Langkah 6: Hapus	22
Prasyarat	24
Mendaftar untuk Akun AWS	24
Memulai	25
Langkah 1: Membuat fungsi Lambda	26
Langkah 2: Buat API HTTP	26
Langkah 3: Uji API Anda	27
(Opsional) Langkah 4: Bersihkan	28

Langkah selanjutnya	29
Tutorial dan lokakarya	31
Tutorial REST API	32
Pilih tutorial integrasi Lambda	32
Tutorial: Buat REST API dengan mengimpor contoh	56
Pilih tutorial integrasi HTTP	66
Tutorial: Buat REST API dengan integrasi pribadi	80
Tutorial: Buat REST API dengan AWS integrasi	86
Tutorial: Buat REST API kalkulator dengan tiga integrasi	92
Tutorial: Buat REST API sebagai proxy Amazon S3	119
Tutorial: Buat REST API sebagai proxy Amazon Kinesis	163
Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI	208
Tutorial: Membuat REST API pribadi	242
HTTP API tutorial	249
Tutorial: Bangun API HTTP CRUD dengan Lambda dan DynamoDB	249
Tutorial: Buat API HTTP dengan integrasi pribadi ke Amazon ECS	262
WebSocket Tutorial API	269
Tutorial: Membuat aplikasi WebSocket obrolan	270
Tutorial: Membuat WebSocket API dengan AWS integrasi	275
API Gateway REST APIs	288
Mengembangkan	288
Contoh sumber daya untuk REST API	289
Fitur REST API tambahan untuk pengembangan	292
Jenis titik akhir API Gateway	294
Kebijakan Keamanan	298
Jenis alamat IP untuk REST APIs di API Gateway	317
Metode	319
Kontrol akses	339
Integrasi	426
Minta validasi	520
Transformasi data	550
Tanggapan Gateway	632
CORS	643
Jenis media biner	657
Panggil	689
OpenAPI	722

Publikasikan	737
Menyebarkan REST APIs	738
Nama domain kustom	784
Pengoptimalan	856
Pengaturan cache	856
Pengkodean konten	869
Mendistribusikan	875
Paket penggunaan	876
Dokumentasi API	904
Generasi SDK	969
Jual APIs sebagai SaaS Anda	995
Lindungi	1000
TLS timbal balik	1001
Sertifikat Klien	1008
AWS WAF	1048
Throttling	1051
REST pribadi APIs	1054
Memantau	1111
CloudWatch metrik	1112
CloudWatch log	1121
Firehose	1127
Variabel untuk pencatatan akses untuk API Gateway	1128
X-Ray	1141
Portal API Gateway	1155
Pertimbangan-pertimbangan	1155
Produk portal	1156
Contoh produk portal adopsi hewan peliharaan	1156
Langkah selanjutnya	1158
Buat produk portal	1159
Buat titik akhir REST produk	1161
Buat halaman produk	1163
Perbarui produk portal	1164
Memperbarui titik akhir REST produk	1165
Perbarui halaman produk	1166
Bagikan produk portal	1166
Aktifkan coba untuk titik akhir REST produk di portal Anda	1172

Hapus produk portal	1173
Buat portal	1174
Pilih nama domain	1174
Buat portal	1175
Perbarui portal	1177
Perbarui portal	1177
Pratinjau portal	1178
Pertimbangan-pertimbangan	1178
Pratinjau portal	1178
Publikasikan portal	1179
Pertimbangan-pertimbangan	1179
Publikasikan portal	1179
Gunakan portal	1180
Gunakan portal	1180
Memecahkan masalah portal Anda	1180
Nonaktifkan portal	1181
Nonaktifkan portal	1181
Hapus portal	1182
Hapus portal	1182
API Gateway HTTP APIs	1183
Mengembangkan	1183
Membuat API HTTP	1184
Rute	1185
Jenis alamat IP untuk HTTP APIs di API Gateway	1188
Kontrol akses	1190
Integrasi	1209
CORS	1228
Pemetaan parameter	1231
OpenAPI	1238
Publikasikan	1248
Tahapan	1249
Kebijakan keamanan untuk HTTP APIs di API Gateway	1251
Nama domain kustom	1253
Lindungi	1262
Throttling	1263
TLS timbal balik	1264

Memantau	1271
Metrik	1271
Pencatatan log	1274
Pemecahan Masalah	1284
Integrasi Lambda	1284
Pengotorisasi JWT	1287
API Gateway WebSocket APIs	1289
Ikhtisar WebSocket APIs	1289
Menggunakan rute untuk WebSocket API	1290
Mengirim data ke aplikasi klien yang terhubung	1290
WebSocket Kode status API	1291
Mengelola pengguna yang terhubung dan aplikasi klien	1292
Panggil integrasi backend Anda	1295
WebSocket ekspresi seleksi	1299
Mengembangkan	1307
Buat dan konfigurasi	1308
Jenis alamat IP untuk WebSocket APIs di API Gateway	1310
Rute	1311
Kontrol akses	1320
Integrasi	1328
Minta validasi	1338
Transformasi data	1341
Jenis media biner	1353
Panggil	1353
Publikasikan	1357
Tahapan	1357
Menerapkan API WebSocket	1360
Kebijakan keamanan untuk WebSocket APIs di API Gateway	1362
Nama domain kustom	1364
Lindungi	1372
Pelambatan tingkat akun per Wilayah	1372
Pelambatan tingkat rute	1373
Memantau	1373
Metrik-metrik	1374
Logging	1376
API Gateway ARNs	1385

Sumber daya API dan WebSocket API HTTP	1385
Sumber daya REST API	1388
Sumber nama domain	1392
execute-api(HTTP APIs, WebSocket APIs, dan REST APIs)	1393
Ekstensi OpenAPI	1394
x-amazon-apigateway-any-method	1395
x-amazon-apigateway-any-contoh metode	1396
x-amazon-apigateway-cors	1397
x-amazon-apigateway-cors contoh	1397
x-amazon-apigateway-api-key-source	1398
x-amazon-apigateway-api-contoh sumber kunci	1399
x-amazon-apigateway-auth	1400
x-amazon-apigateway-auth contoh	1400
x-amazon-apigateway-authorizer	1401
x-amazon-apigateway-authorizer contoh untuk REST APIs	1404
x-amazon-apigateway-authorizer contoh untuk HTTP APIs	1408
x-amazon-apigateway-authtype	1410
x-amazon-apigateway-authtype contoh	1410
Lihat juga	1412
x-amazon-apigateway-binary-tipe media	1412
x-amazon-apigateway-binary-media-jenis contoh	1412
x-amazon-apigateway-documentation	1412
x-amazon-apigateway-documentation contoh	1413
x-amazon-apigateway-endpoint-modus akses	1414
Contoh x-amazon-apigateway-endpoint-access-mode	1414
x-amazon-apigateway-endpoint-konfigurasi	1414
x-amazon-apigateway-endpoint-contoh konfigurasi	1415
x-amazon-apigateway-gateway-tanggapan	1416
x-amazon-apigateway-gateway-contoh tanggapan	1416
x-amazon-apigateway-gateway-Responses.gatewayResponse	1417
x-amazon-apigateway-gateway-Responses.gatewayResponse contoh	1417
x-amazon-apigateway-gateway-Responses.ResponseParameters	1418
x-amazon-apigateway-gateway-Responses.ResponseParameters contoh	1418
x-amazon-apigateway-gateway-Responses.ResponseTemplates	1419
x-amazon-apigateway-gateway-Responses.responseTemplates contoh	1419
x-amazon-apigateway-importexport-versi	1420

x-amazon-apigateway-importexport-versi contoh	1420
x-amazon-apigateway-integration	1420
x-amazon-apigateway-integration contoh	1427
x-amazon-apigateway-integrations	1429
x-amazon-apigateway-integrations contoh	1429
x-amazon-apigateway-integration.requestTemplates	1431
x-amazon-apigateway-integrationContoh. RequestTemplates	1431
x-amazon-apigateway-integration.requestParameters	1432
Contoh x-amazon-apigateway-integration.requestParameters	1433
x-amazon-apigateway-integration.tanggapan	1434
Contoh x-amazon-apigateway-integration.responses	1435
x-amazon-apigateway-integration.respon	1436
Contoh x-amazon-apigateway-integration.response	1437
x-amazon-apigateway-integration.ResponseTemplates	1437
x-amazon-apigateway-integrationContoh .responseTemplate	1438
x-amazon-apigateway-integration.ResponseParameters	1438
Contoh x-amazon-apigateway-integration.responseParameters	1439
x-amazon-apigateway-integration.tlsConfig	1439
x-amazon-apigateway-integrationContoh .tlsConfig	1441
x-amazon-apigateway-minimum-ukuran kompresi	1442
x-amazon-apigateway-minimum-contoh ukuran kompresi	1442
x-amazon-apigateway-policy	1442
Contoh x-amazon-apigateway-policy	1442
x-amazon-apigateway-request-validator	1443
Contoh x-amazon-apigateway-request-validator	1443
x-amazon-apigateway-request-validator	1444
Contoh x-amazon-apigateway-request-validators	1445
x-amazon-apigateway-request-Validators.RequestValidator	1445
Contoh x-amazon-apigateway-request-validators.requestValidator	1446
x-amazon-apigateway-security-kebijakan	1446
Contoh x-amazon-apigateway-security-policy	1446
x-amazon-apigateway-tag-nilai	1446
Contoh x-amazon-apigateway-tag-value	1447
Keamanan	1448
Perlindungan data	1449
Enkripsi data	1450

Privasi lalu lintas jaringan internet	1452
Manajemen identitas dan akses	1452
Audiens	1453
Mengautentikasi dengan identitas	1453
Mengelola akses menggunakan kebijakan	1454
Cara kerja Amazon API Gateway dengan IAM	1456
Contoh kebijakan berbasis identitas	1462
Contoh kebijakan berbasis sumber daya	1474
Pemecahan masalah	1474
Menggunakan Peran Terkait Layanan	1476
Pencatatan dan pemantauan	1481
Bekerja dengan CloudTrail	1483
Bekerja dengan AWS Config	1486
Validasi kepatuhan	1489
Ketahanan	1490
Keamanan infrastruktur	1490
Konfigurasi dan analisis kerentanan	1491
Praktik terbaik	1491
Penandaan	1494
Sumber daya API Gateway yang dapat ditandai	1495
Warisan tag di Amazon API Gateway V1	1496
Pembatasan tag dan konvensi penggunaan	1497
Kontrol Akses Berbasis Atribut	1498
Batasi tindakan berdasarkan tag sumber daya	1498
Izinkan tindakan berdasarkan tag sumber daya	1499
Tolak operasi penandaan	1500
Izinkan operasi penandaan	1501
Referensi API	1503
Titik akhir layanan API Gateway	1503
IPv4 titik akhir	1503
Titik akhir Dualstack (IPv4 dan IPv6)	1504
Menentukan titik akhir	1504
Menggunakan IPv6 alamat dalam kebijakan IAM	1505
Kuota dan catatan penting	1507
Kuota tingkat akun API Gateway, per Wilayah	1507
Kuota API Gateway untuk membuat, menerapkan, dan mengelola API	1508

Kuota REST API	1511
Praktik terbaik untuk mengurangi kuota Anda	1518
Kuota HTTP API	1519
Praktik terbaik untuk mengurangi kuota Anda	1522
Kuota portal	1522
WebSocket Kuota API	1523
Catatan penting	1526
Catatan penting Amazon API Gateway untuk HTTP APIs	1526
Catatan penting Amazon API Gateway untuk HTTP dan WebSocket APIs	1526
Catatan penting untuk REST APIs dan WebSocket APIs	1526
Catatan penting untuk WebSocket APIs	1527
Catatan penting untuk REST APIs	1527
Riwayat dokumen	1534
Pembaruan lebih awal	1548
.....	mdlix

Apa itu Amazon API Gateway?

Amazon API Gateway adalah AWS layanan untuk membuat, menerbitkan, memelihara, memantau, dan mengamankan REST, HTTP, dan WebSocket APIs pada skala apa pun. Pengembang API dapat membuat akses APIs itu AWS atau layanan web lainnya, serta data yang disimpan di [AWS Cloud](#). Sebagai pengembang API Gateway API, Anda dapat membuat APIs untuk digunakan dalam aplikasi klien Anda sendiri. Atau Anda dapat APIs menyediakannya untuk pengembang aplikasi pihak ketiga. Untuk informasi selengkapnya, lihat [the section called “Siapa yang menggunakan API Gateway?”](#).

API Gateway membuat RESTful APIs bahwa:

- Berbasis HTTP.
- Aktifkan komunikasi client-server stateless.
- Menerapkan metode HTTP standar seperti GET, POST, PUT, PATCH, dan DELETE.

Untuk informasi selengkapnya tentang API Gateway REST APIs dan HTTP APIs [the section called “Pilih antara REST APIs dan HTTP APIs”](#), lihat [API Gateway HTTP APIs](#), [the section called “Gunakan API Gateway untuk membuat REST APIs”](#), dan [the section called “Mengembangkan”](#).

API Gateway membuat WebSocket APIs bahwa:

- Patuhi [WebSocket](#) protokol, yang memungkinkan komunikasi full-duplex stateful antara klien dan server.
- Rutekan pesan masuk berdasarkan konten pesan.

Untuk informasi selengkapnya tentang API Gateway WebSocket APIs, lihat [the section called “Gunakan API Gateway untuk membuat WebSocket APIs”](#) dan [the section called “Ikhtisar WebSocket APIs”](#).

Topik

- [Arsitektur API Gateway](#)
- [Fitur API Gateway](#)
- [Kasus penggunaan API Gateway](#)
- [Mengakses API Gateway](#)
- [Bagian dari infrastruktur AWS tanpa server](#)

- [Cara memulai dengan Amazon API Gateway](#)
- [Konsep Amazon API Gateway](#)
- [Pilih antara REST APIs dan HTTP APIs](#)
- [Memulai dengan konsol REST API](#)

Arsitektur API Gateway

Diagram berikut menunjukkan arsitektur API Gateway.

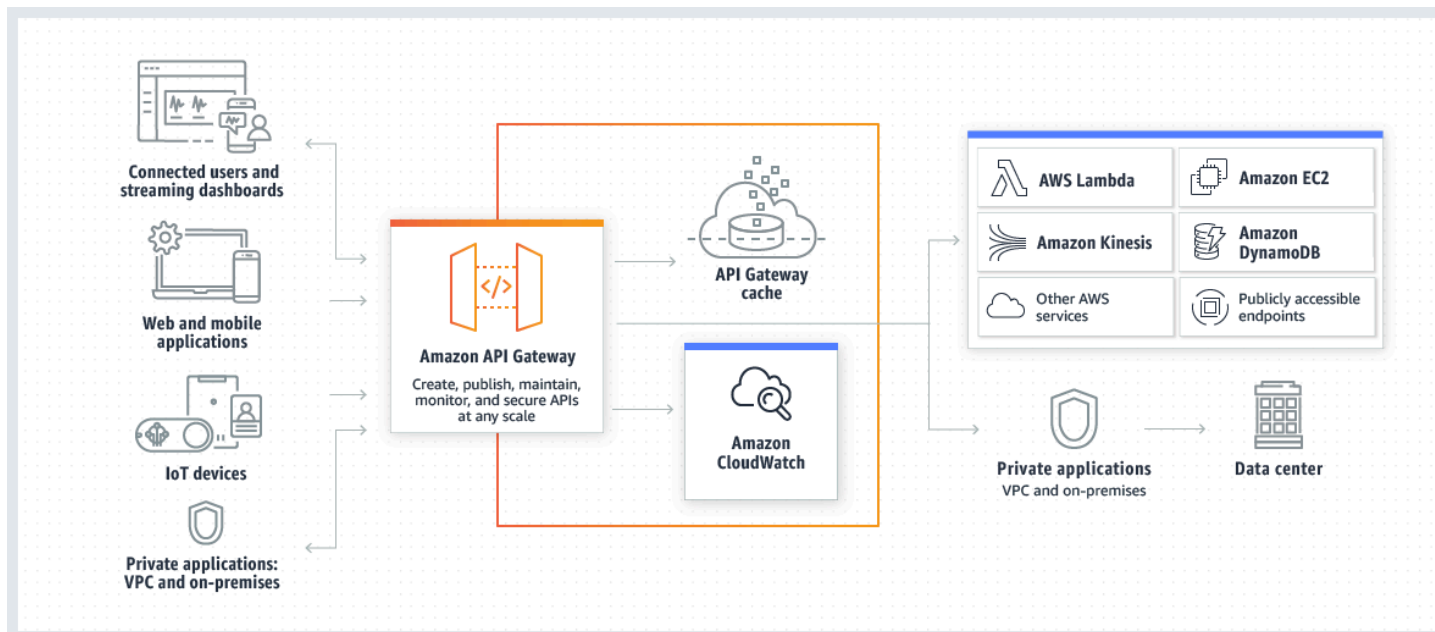


Diagram ini menggambarkan bagaimana APIs Anda membangun di Amazon API Gateway memberi Anda atau pelanggan pengembang Anda pengalaman pengembang yang terintegrasi dan konsisten untuk membangun aplikasi tanpa AWS server. API Gateway menangani semua tugas yang terlibat dalam menerima dan memproses hingga ratusan ribu panggilan API bersamaan. Tugas-tugas ini termasuk manajemen lalu lintas, otorisasi dan kontrol akses, pemantauan, dan manajemen versi API.

API Gateway bertindak sebagai “pintu depan” bagi aplikasi untuk mengakses data, logika bisnis, atau fungsionalitas dari layanan backend Anda, seperti beban kerja yang berjalan di Amazon Elastic Compute Cloud (Amazon EC2), kode yang berjalan, aplikasi web apa pun, atau aplikasi komunikasi AWS Lambda real-time.

Fitur API Gateway

Amazon API Gateway menawarkan fitur-fitur seperti berikut:

- [Support untuk stateful \(WebSocket\) dan stateless \(HTTP dan REST\). APIs](#)
- Mekanisme [otentikasi](#) yang kuat dan fleksibel, seperti AWS Identity and Access Management kebijakan, fungsi otorisasi Lambda, dan kumpulan pengguna Amazon Cognito.
- [Penerapan rilis Canary untuk meluncurkan perubahan](#) dengan aman.
- [CloudTrail](#) logging dan pemantauan penggunaan API dan perubahan API.
- CloudWatch akses logging dan eksekusi logging, termasuk kemampuan untuk mengatur alarm. Untuk informasi selengkapnya, lihat [the section called “CloudWatch metrik”](#) dan [the section called “Metrik-metrik”](#).
- Kemampuan untuk menggunakan CloudFormation template untuk mengaktifkan pembuatan API. Untuk informasi selengkapnya, lihat [Referensi Jenis Sumber Daya Amazon API Gateway dan Referensi Jenis Sumber Daya Amazon API Gateway V2](#).
- Support untuk [nama domain kustom](#).
- Integrasi dengan [AWS WAF](#) untuk APIs melindungi Anda dari eksploitasi web umum.
- Integrasi dengan [AWS X-Ray](#) untuk memahami dan memprioritaskan latensi kinerja.

Untuk daftar lengkap rilis fitur API Gateway, lihat [Riwayat dokumen](#).

Kasus penggunaan API Gateway

Bagian kasus penggunaan berikut menyajikan ikhtisar tentang berbagai jenis API Gateway APIs dan berbagai jenis pengembang yang menggunakan API Gateway. Untuk informasi lebih rinci tentang perbedaan antara REST APIs dan HTTP APIs, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

Topik

- [Gunakan API Gateway untuk membuat REST APIs](#)
- [Gunakan API Gateway untuk membuat HTTP APIs](#)
- [Gunakan API Gateway untuk membuat WebSocket APIs](#)
- [Siapa yang menggunakan API Gateway?](#)

Gunakan API Gateway untuk membuat REST APIs

API Gateway REST terdiri dari sumber daya dan metode. Resource adalah entitas logis yang dapat diakses aplikasi melalui jalur sumber daya. Metode sesuai dengan permintaan REST API yang dikirimkan oleh pengguna API Anda dan respons yang dikembalikan ke pengguna.

Misalnya, `/incomes` bisa menjadi jalur sumber daya yang mewakili pendapatan pengguna aplikasi. Sumber daya dapat memiliki satu atau lebih operasi yang ditentukan oleh kata kerja HTTP yang sesuai seperti GET, POST, PUT, PATCH, dan DELETE. Kombinasi jalur sumber daya dan operasi mengidentifikasi metode API. Misalnya, `POST /incomes` metode dapat menambahkan pendapatan yang diperoleh oleh penelepon, dan `GET /expenses` metode dapat menanyakan biaya yang dilaporkan yang dikeluarkan oleh penelepon.

Aplikasi tidak perlu tahu di mana data yang diminta disimpan dan diambil dari backend. Di API Gateway REST APIs, frontend dikapsulasi oleh permintaan metode dan respons metode. API berinteraksi dengan backend melalui permintaan integrasi dan respons integrasi.

Misalnya, dengan DynamoDB sebagai backend, pengembang API menyiapkan permintaan integrasi untuk meneruskan permintaan metode masuk ke backend yang dipilih. Penyiapan mencakup spesifikasi tindakan DynamoDB yang sesuai, peran dan kebijakan IAM yang diperlukan, dan transformasi data input yang diperlukan. Backend mengembalikan hasilnya ke API Gateway sebagai respons integrasi.

Untuk merutekan respons integrasi ke respons metode yang sesuai (dari kode status HTTP yang diberikan) ke klien, Anda dapat mengonfigurasi respons integrasi untuk memetakan parameter respons yang diperlukan dari integrasi ke metode. Anda kemudian menerjemahkan format data keluaran backend ke format frontend, jika perlu. API Gateway memungkinkan Anda untuk menentukan skema atau model untuk [payload](#) untuk memfasilitasi pengaturan template pemetaan tubuh.

API Gateway menyediakan fungsionalitas manajemen REST API seperti berikut ini:

- Support untuk membuat SDKs dan membuat dokumentasi API menggunakan ekstensi API Gateway ke OpenAPI
- Pelambatan permintaan HTTP

Gunakan API Gateway untuk membuat HTTP APIs

HTTP APIs memungkinkan Anda untuk membuat RESTful APIs dengan latensi yang lebih rendah dan biaya lebih rendah daripada REST APIs.

Anda dapat menggunakan HTTP APIs untuk mengirim permintaan ke AWS Lambda fungsi atau ke titik akhir HTTP yang dapat dirutekan secara publik.

Misalnya, Anda dapat membuat API HTTP yang terintegrasi dengan fungsi Lambda di backend. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien.

HTTP APIs mendukung [OpenID Connect dan otorisasi OAuth 2.0](#). Mereka datang dengan dukungan bawaan untuk berbagi sumber daya lintas asal (CORS) dan penerapan otomatis.

Untuk mempelajari selengkapnya, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

Gunakan API Gateway untuk membuat WebSocket APIs

Dalam WebSocket API, klien dan server dapat saling mengirim pesan kapan saja. Server backend dapat dengan mudah mendorong data ke pengguna dan perangkat yang terhubung, menghindari kebutuhan untuk menerapkan mekanisme pemungutan suara yang kompleks.

Misalnya, Anda dapat membuat aplikasi tanpa server menggunakan API Gateway WebSocket API dan mengirim serta menerima pesan AWS Lambda ke dan dari pengguna individu atau grup pengguna di ruang obrolan. Atau Anda dapat memanggil layanan backend seperti, Amazon AWS Lambda Kinesis, atau titik akhir HTTP berdasarkan konten pesan.

Anda dapat menggunakan API Gateway WebSocket APIs untuk membangun aplikasi komunikasi real-time yang aman tanpa harus menyediakan atau mengelola server apa pun untuk mengelola koneksi atau pertukaran data skala besar. Kasus penggunaan yang ditargetkan mencakup aplikasi waktu nyata seperti berikut ini:

- Aplikasi obrolan
- Dasbor real-time seperti ticker saham
- Peringatan dan notifikasi waktu nyata

API Gateway menyediakan fungsionalitas manajemen WebSocket API seperti berikut ini:

- Pemantauan dan pembatasan koneksi dan pesan

- Menggunakan AWS X-Ray untuk melacak pesan saat mereka melakukan perjalanan melalui layanan APIs to backend
- Integrasi mudah dengan titik HTTP/HTTPS akhir

Siapa yang menggunakan API Gateway?

Ada dua jenis pengembang yang menggunakan API Gateway: pengembang API dan pengembang aplikasi.

Pengembang API membuat dan menerapkan API untuk mengaktifkan fungsionalitas yang diperlukan di API Gateway. Pengembang API harus menjadi pengguna di AWS akun yang memiliki API.

Pengembang aplikasi membangun aplikasi yang berfungsi untuk memanggil AWS layanan dengan menjalankan API REST WebSocket atau yang dibuat oleh pengembang API di API Gateway.

Pengembang aplikasi adalah pelanggan pengembang API. [Pengembang aplikasi tidak perlu memiliki AWS akun, asalkan API tidak memerlukan izin IAM atau mendukung otorisasi pengguna melalui penyedia identitas federasi pihak ketiga yang didukung oleh federasi identitas kumpulan pengguna Amazon Cognito](#). Penyedia identitas tersebut termasuk Amazon, kumpulan pengguna Amazon Cognito, Facebook, dan Google.

Membuat dan mengelola API Gateway API

Pengembang API bekerja dengan komponen layanan API Gateway untuk manajemen API, bernamaapigateway, untuk membuat, mengonfigurasi, dan menerapkan API.

Sebagai pengembang API, Anda dapat membuat dan mengelola API dengan menggunakan konsol API Gateway, yang dijelaskan dalam [Memulai dengan API Gateway](#), atau dengan memanggil [Referensi API](#). Ada beberapa cara untuk memanggil API ini. Mereka termasuk menggunakan AWS Command Line Interface (AWS CLI), atau dengan menggunakan AWS SDK. Selain itu, Anda dapat mengaktifkan pembuatan API dengan [AWS CloudFormation templat](#) atau (dalam kasus REST APIs dan HTTP APIs) [Ekstensi OpenAPI untuk API Gateway](#).

Untuk daftar Wilayah tempat API Gateway tersedia, serta titik akhir layanan kontrol terkait, lihat Titik Akhir [dan Kuota Amazon API Gateway](#).

Memanggil API Gateway API

Pengembang aplikasi bekerja dengan komponen layanan API Gateway untuk eksekusi API, bernamaexecute-api, untuk memanggil API yang dibuat atau diterapkan di API Gateway. Entitas

pemrograman yang mendasarinya diekspos oleh API yang dibuat. Ada beberapa cara untuk memanggil API semacam itu. Untuk mempelajari selengkapnya, lihat [Panggil REST APIs di API Gateway](#) dan [Memohon WebSocket APIs](#).

Mengakses API Gateway

Anda dapat mengakses Amazon API Gateway dengan cara berikut:

- **Konsol Manajemen AWS**— Konsol Manajemen AWS Menyediakan antarmuka web untuk membuat dan mengelola APIs. Setelah menyelesaikan langkah-langkahnya [Prasyarat](#), Anda dapat mengakses konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
- **AWS SDKs**— Jika Anda menggunakan bahasa pemrograman yang AWS menyediakan SDK, Anda dapat menggunakan SDK untuk mengakses API Gateway. SDKs menyederhanakan otentikasi, mengintegrasikan dengan mudah dengan lingkungan pengembangan Anda, dan menyediakan akses ke perintah API Gateway. Untuk informasi lebih lanjut, lihat [Alat untuk Amazon Web Services](#).
- **API Gateway V1 dan V2 APIs** — Jika Anda menggunakan bahasa pemrograman yang tidak tersedia untuk SDK, lihat Referensi [API Amazon API Gateway Versi 1](#) dan Referensi [API Amazon API Gateway Versi 2](#).
- **AWS Command Line Interface** – Untuk informasi selengkapnya, lihat [Menyiapkan dengan AWS Command Line Interface](#) di AWS Command Line Interface Panduan Pengguna.
- **AWS Tools for Windows PowerShell**— Untuk informasi selengkapnya, lihat [Menyiapkan AWS Tools for Windows PowerShell](#) di Panduan Alat AWS untuk PowerShell Pengguna.

Bagian dari infrastruktur AWS tanpa server

Bersama dengan [AWS Lambda](#), API Gateway membentuk bagian yang menghadap aplikasi dari infrastruktur tanpa AWS server. Untuk mempelajari lebih lanjut tentang memulai dengan tanpa server, lihat Panduan Pengembang Tanpa [Server](#).

Agar aplikasi dapat memanggil AWS layanan yang tersedia untuk umum, Anda dapat menggunakan Lambda untuk berinteraksi dengan layanan yang diperlukan dan mengekspos fungsi Lambda melalui metode API di API Gateway. AWS Lambda menjalankan kode Anda pada infrastruktur komputasi yang sangat tersedia. Ini melakukan eksekusi dan administrasi sumber daya komputasi yang diperlukan. Untuk mengaktifkan aplikasi tanpa server, API Gateway mendukung [integrasi proxy yang efisien](#) dengan AWS Lambda dan titik akhir HTTP.

Cara memulai dengan Amazon API Gateway

Untuk pengenalan Amazon API Gateway, lihat berikut ini:

- [Memulai](#), yang menyediakan panduan untuk membuat API HTTP.
- [Tanah tanpa server](#), yang menyediakan video instruksional.
- [Happy Little API Shorts](#), yang merupakan serangkaian video instruksional singkat.

Konsep Amazon API Gateway

Bagian berikut menjelaskan konsep pengantar untuk menggunakan API Gateway.

API Gateway

API Gateway adalah AWS layanan yang mendukung hal-hal berikut:

- Membuat, menyebarkan, dan mengelola antarmuka pemrograman [RESTful](#) aplikasi (API) untuk mengekspos titik akhir HTTP backend, AWS Lambda fungsi, atau layanan lainnya. AWS
- Membuat, menerapkan, dan mengelola [WebSocket](#) API untuk mengekspos AWS Lambda fungsi atau layanan lainnya AWS .
- Memanggil metode API terbuka melalui HTTP frontend dan WebSocket titik akhir.

API Gateway API REST

Kumpulan sumber daya HTTP dan metode yang terintegrasi dengan endpoint HTTP backend, fungsi Lambda, atau layanan lainnya. AWS Anda dapat menyebarkan koleksi ini dalam satu atau lebih tahap. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Setiap sumber daya API dapat mengekspos satu atau beberapa metode API yang memiliki kata kerja HTTP unik yang didukung oleh API Gateway. Untuk informasi selengkapnya, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

API Gateway API HTTP

Kumpulan rute dan metode yang terintegrasi dengan titik akhir HTTP backend atau fungsi Lambda. Anda dapat menyebarkan koleksi ini dalam satu atau lebih tahap. Setiap rute dapat mengekspos satu atau beberapa metode API yang memiliki kata kerja HTTP unik yang didukung oleh API Gateway. Untuk informasi selengkapnya, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

API Gateway WebSocket API

Kumpulan WebSocket rute dan kunci rute yang terintegrasi dengan titik akhir HTTP backend, fungsi Lambda, atau layanan lainnya. AWS Anda dapat menyebarkan koleksi ini dalam satu atau lebih tahap. Metode API dipanggil melalui WebSocket koneksi frontend yang dapat Anda kaitkan dengan nama domain kustom terdaftar.

Penerapan API

point-in-timeCuplikan API Gateway API Anda. Agar tersedia bagi klien untuk digunakan, penerapan harus dikaitkan dengan satu atau beberapa tahapan API.

Pengembang API

AWS Akun Anda yang memiliki penerapan API Gateway (misalnya, penyedia layanan yang juga mendukung akses terprogram).

Titik akhir API

Nama host untuk API di API Gateway yang diterapkan ke Wilayah tertentu. Nama host adalah dari formulir `{api-id}.execute-api.{region}.amazonaws.com`. Jenis endpoint API berikut didukung:

- [Titik akhir API yang dioptimalkan tepi](#)
- [Titik akhir API pribadi](#)
- [Titik akhir API regional](#)

Kunci API

String alfanumerik yang digunakan API Gateway untuk mengidentifikasi pengembang aplikasi yang menggunakan REST atau API Anda. WebSocket API Gateway dapat menghasilkan kunci API atas nama Anda, atau Anda dapat mengimpornya dari file CSV. Anda dapat menggunakan kunci API bersama dengan [otorisasi Lambda](#) atau [paket penggunaan](#) untuk mengontrol akses ke Anda. APIs

Lihat [titik akhir API](#).

Pemilik API

Lihat [pengembang API](#).

Tahap API

Referensi logis ke status siklus hidup API Anda (misalnya, 'dev', 'prod', 'beta', 'v2'). Tahapan API diidentifikasi oleh ID API dan nama panggung.

Pengembang aplikasi

Pembuat aplikasi yang mungkin atau mungkin tidak memiliki AWS akun dan berinteraksi dengan API yang Anda, pengembang API, telah diterapkan. Pengembang aplikasi adalah pelanggan Anda. Pengembang aplikasi biasanya diidentifikasi oleh [kunci API](#).

URL panggilan balik

Ketika klien baru terhubung melalui WebSocket koneksi, Anda dapat memanggil integrasi di API Gateway untuk menyimpan URL callback klien. Anda kemudian dapat menggunakan URL callback tersebut untuk mengirim pesan ke klien dari sistem backend.

Portal pengembang

Aplikasi tempat penyedia API dapat membagikan dokumentasi mereka APIs dan API kepada konsumen API. APIs dikelompokkan ke dalam produk, yang merupakan kumpulan titik akhir REST API, dokumentasi API, dan dokumentasi produk tambahan.

Lihat [Portal API Gateway](#).

Titik akhir API yang dioptimalkan tepi

Nama host default API Gateway API yang diterapkan ke Wilayah tertentu saat menggunakan CloudFront distribusi untuk memfasilitasi akses klien biasanya dari seluruh AWS Wilayah. Permintaan API dirutekan ke CloudFront Point of Presence (POP) terdekat, yang biasanya meningkatkan waktu koneksi untuk klien yang beragam secara geografis.

Lihat [titik akhir API](#).

Permintaan integrasi

Antarmuka internal rute WebSocket API atau metode REST API di API Gateway, di mana Anda memetakan isi permintaan rute atau parameter dan isi permintaan metode ke format yang diperlukan oleh backend.

Respon integrasi

Antarmuka internal rute WebSocket API atau metode REST API di API Gateway, tempat Anda memetakan kode status, header, dan payload yang diterima dari backend ke format respons yang dikembalikan ke aplikasi klien.

Template pemetaan

Skrip dalam [Velocity Template Language \(VTL\)](#) yang mengubah badan permintaan dari format data frontend ke format data backend, atau yang mengubah badan respons dari format data

backend ke format data frontend. Template pemetaan dapat ditentukan dalam permintaan integrasi atau dalam respons integrasi. Mereka dapat mereferensikan data yang tersedia saat runtime sebagai variabel konteks dan tahap.

Pemetaan dapat sesederhana [transformasi identitas](#) yang melewati header atau badan melalui integrasi apa adanya dari klien ke backend untuk permintaan. Hal yang sama berlaku untuk respons, di mana muatan diteruskan dari backend ke klien.

Permintaan metode

Antarmuka publik metode API di API Gateway yang menentukan parameter dan isi yang harus dikirimkan oleh pengembang aplikasi dalam permintaan untuk mengakses backend melalui API.

Metode respon

Antarmuka publik REST API yang mendefinisikan kode status, header, dan model tubuh yang diharapkan oleh pengembang aplikasi dalam respons dari API.

Integrasi tiruan

Dalam integrasi tiruan, respons API dihasilkan dari API Gateway secara langsung, tanpa perlu backend integrasi. Sebagai pengembang API, Anda memutuskan bagaimana API Gateway merespons permintaan integrasi tiruan. Untuk ini, Anda mengonfigurasi permintaan integrasi metode dan respons integrasi untuk mengaitkan respons dengan kode status tertentu.

Model

Skema data yang menentukan struktur data dari permintaan atau muatan respons. Model diperlukan untuk menghasilkan SDK API yang diketik dengan kuat. Ini juga digunakan untuk memvalidasi muatan. Sebuah model nyaman untuk menghasilkan template pemetaan sampel untuk memulai pembuatan template pemetaan produksi. Meskipun berguna, model tidak diperlukan untuk membuat template pemetaan.

Portal

Lihat [Portal pengembang](#).

Produk Portal

Layanan atau fungsionalitas yang ingin Anda bagikan. Produk portal Anda adalah kumpulan titik akhir REST produk dan halaman produk. Titik akhir REST produk adalah titik akses ke produk portal Anda, dan mereka terdiri dari jalur dan metode REST API dan tahap penerapannya. Halaman produk adalah dokumentasi yang Anda berikan untuk menjelaskan bagaimana konsumen API dapat menggunakan titik akhir produk Anda. Anda dapat berbagi produk di seluruh AWS akun untuk menambahkannya ke portal.

Lihat [Membuat produk](#).

API Pribadi

Lihat [Titik akhir API pribadi](#).

Titik akhir API pribadi

Titik akhir API yang diekspos melalui titik akhir VPC antarmuka dan memungkinkan klien mengakses sumber daya API pribadi dengan aman di dalam VPC. Private APIs diisolasi dari internet publik, dan mereka hanya dapat diakses menggunakan titik akhir VPC untuk API Gateway yang telah diberikan akses.

Integrasi pribadi

Jenis integrasi API Gateway bagi klien untuk mengakses sumber daya di dalam VPC pelanggan melalui titik akhir REST API pribadi tanpa mengekspos sumber daya ke internet publik.

Integrasi proxy

Konfigurasi integrasi API Gateway yang disederhanakan. Anda dapat mengatur integrasi proxy sebagai integrasi proxy HTTP atau integrasi proxy Lambda.

Untuk integrasi proxy HTTP, API Gateway meneruskan seluruh permintaan dan respons antara frontend dan backend HTTP. Untuk integrasi proxy Lambda, API Gateway mengirimkan seluruh permintaan sebagai input ke fungsi Lambda backend. API Gateway kemudian mengubah output fungsi Lambda menjadi respons HTTP frontend.

Dalam REST APIs, integrasi proxy paling sering digunakan dengan sumber daya proxy, yang diwakili oleh variabel jalur serakah (misalnya, {proxy+}) dikombinasikan dengan metode catch-allANY.

Buat cepat

Anda dapat menggunakan quick create untuk menyederhanakan pembuatan HTTP API. Pembuatan cepat membuat API dengan integrasi Lambda atau HTTP, rute tangkapan semua default, dan tahap default yang dikonfigurasi untuk menerapkan perubahan secara otomatis. Untuk informasi selengkapnya, lihat [the section called “Buat API HTTP dengan menggunakan AWS CLI”](#).

Titik akhir API regional

Nama host API yang disebar ke Wilayah tertentu dan dimaksudkan untuk melayani klien, seperti EC2 instance, di Wilayah yang sama AWS. Permintaan API ditargetkan langsung ke API

Gateway API khusus Wilayah tanpa melalui distribusi apa pun CloudFront . Untuk permintaan In-Region, titik akhir Regional melewati perjalanan pulang pergi yang tidak perlu ke distribusi CloudFront

Selain itu, Anda dapat menerapkan [perutean berbasis latensi](#) pada titik akhir Regional untuk menerapkan API ke beberapa Wilayah menggunakan konfigurasi titik akhir API Regional yang sama, menetapkan nama domain kustom yang sama untuk setiap API yang diterapkan, dan mengonfigurasi catatan DNS berbasis latensi di Route 53 untuk merutekan permintaan klien ke Wilayah yang memiliki latensi terendah.

Lihat [titik akhir API](#).

Rute

WebSocket Rute di API Gateway digunakan untuk mengarahkan pesan masuk ke integrasi tertentu, seperti AWS Lambda fungsi, berdasarkan konten pesan. Saat menentukan WebSocket API, Anda menentukan kunci rute dan backend integrasi. Kunci rute adalah atribut di badan pesan. Ketika kunci rute dicocokkan dalam pesan masuk, backend integrasi dipanggil.

Rute default juga dapat diatur untuk kunci rute yang tidak cocok atau untuk menentukan model proxy yang meneruskan pesan melalui apa adanya ke komponen backend yang melakukan perutean dan memproses permintaan.

Permintaan rute

Antarmuka publik metode WebSocket API di API Gateway yang mendefinisikan isi yang harus dikirimkan oleh pengembang aplikasi dalam permintaan untuk mengakses backend melalui API.

Respon rute

Antarmuka publik WebSocket API yang mendefinisikan kode status, header, dan model body yang diharapkan pengembang aplikasi dari API Gateway.

Rencana penggunaan

[Paket penggunaan](#) menyediakan klien API terpilih dengan akses ke satu atau beberapa REST yang diterapkan atau WebSocket APIs. Anda dapat menggunakan paket penggunaan untuk mengonfigurasi pembatasan dan batas kuota, yang diberlakukan pada kunci API klien individual.

WebSocket koneksi

API Gateway mempertahankan koneksi persisten antara klien dan API Gateway itu sendiri. Tidak ada koneksi persisten antara API Gateway dan integrasi backend seperti fungsi Lambda. Layanan backend dipanggil sesuai kebutuhan, berdasarkan konten pesan yang diterima dari klien.

Pilih antara REST APIs dan HTTP APIs

REST APIs dan HTTP APIs keduanya adalah produk RESTful API. REST APIs mendukung lebih banyak fitur daripada HTTP APIs, sedangkan HTTP APIs dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. Pilih REST APIs jika Anda memerlukan fitur seperti kunci API, pembatasan per klien, validasi permintaan, AWS WAF integrasi, atau titik akhir API pribadi. Pilih HTTP APIs jika Anda tidak memerlukan fitur yang disertakan dengan REST APIs.

Bagian berikut merangkum fitur inti yang tersedia di REST APIs dan HTTP APIs. Bila perlu, tautan tambahan disediakan untuk menavigasi antara bagian REST API dan HTTP API dari Panduan Pengembang API Gateway.

Tipe titik akhir

Jenis titik akhir mengacu pada titik akhir yang dibuat API Gateway untuk API Anda. Untuk informasi selengkapnya, lihat [the section called “Jenis titik akhir API Gateway”](#).

Jenis titik akhir	API REST	API HTTPS
Dioptimalkan tepi	Ya	Tidak
Regional	Ya	Ya
Pribadi	Ya	Tidak

Keamanan

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti aktor jahat atau lonjakan lalu lintas. Untuk mempelajari selengkapnya, lihat [the section called “Lindungi”](#) dan [the section called “Lindungi”](#).

Fitur keamanan	API REST	API HTTPS
Autentikasi TLS bersama	Ya	Ya
Sertifikat untuk otentikasi backend	Ya	Tidak

Fitur keamanan	API REST	API HTTPS
AWS WAF	Ya	Tidak

Otorisasi

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Kontrol akses”](#) dan [the section called “Kontrol akses”](#).

Opsi otorisasi	API REST	API HTTPS
IAM	Ya	Ya
Kebijakan sumber daya	Ya	Tidak
Amazon Cognito	Ya	Ya ¹
Otorisasi khusus dengan suatu fungsi AWS Lambda	Ya	Ya
Token Web JSON (JWT) ²	Tidak	Ya

¹ Anda dapat menggunakan Amazon Cognito dengan otorisasi [JWT](#).

² Anda dapat menggunakan [otorisasi Lambda](#) untuk memvalidasi JWTs REST APIs

Manajemen API

Pilih REST APIs jika Anda memerlukan kemampuan manajemen API seperti kunci API dan pembatasan tarif per klien. Lihat informasi selengkapnya di [the section called “Mendistribusikan”](#), [the section called “Nama domain kustom”](#), dan [the section called “Nama domain kustom”](#).

Fitur	API REST	API HTTPS
Domain kustom	Ya	Ya

Fitur	API REST	API HTTPS
Kunci API	Ya	Tidak
Pembatasan tarif per klien	Ya	Tidak
Pelambatan penggunaan per klien	Ya	Tidak
Portal pengembang	Ya	Tidak

Pengembangan

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Untuk informasi lebih lanjut, lihat [the section called “Mengembangkan”](#) dan [the section called “Mengembangkan”](#).

Fitur	API REST	API HTTPS
Konfigurasi CORS	Ya	Ya
Doa uji	Ya	Tidak
Caching	Ya	Tidak
Penerapan yang dikendalikan pengguna	Ya	Ya
Penerapan otomatis	Tidak	Ya
Tanggapan gateway khusus	Ya	Tidak
Penerapan rilis kenari	Ya	Tidak
Minta validasi	Ya	Tidak
Minta transformasi parameter	Ya	Ya
Minta transformasi tubuh	Ya	Tidak

Memantau

API Gateway mendukung beberapa opsi untuk mencatat permintaan API dan memantau permintaan Anda APIs. Untuk informasi selengkapnya, lihat [the section called “Memantau”](#) dan [the section called “Memantau”](#).

Fitur	API REST	API HTTPS
CloudWatch Metrik Amazon	Ya	Ya
Akses log ke CloudWatch Log	Ya	Ya
Akses log ke Amazon Data Firehose	Ya	Tidak
Log eksekusi	Ya	Tidak
AWS X-Ray menelusuri	Ya	Tidak

Integrasi

Integrasi menghubungkan API Gateway API Anda ke sumber daya backend. Untuk informasi selengkapnya, lihat [the section called “Integrasi”](#) dan [the section called “Integrasi”](#).

Fitur	API REST	API HTTPS
Titik akhir HTTP publik	Ya	Ya
AWS layanan	Ya	Ya
AWS Lambda fungsi	Ya	Ya
Integrasi pribadi dengan Network Load Balancers	Ya	Ya
Integrasi pribadi dengan Application Load Balancers	Ya	Ya

Fitur	API REST	API HTTPS
Integrasi pribadi dengan AWS Cloud Map	Tidak	Ya
Integrasi tiruan	Ya	Tidak
Streaming respons	Ya	Tidak

Memulai dengan konsol REST API

Dalam latihan memulai ini, Anda membuat REST API tanpa server menggunakan konsol API Gateway REST API. Tanpa server APIs memungkinkan Anda fokus pada aplikasi Anda alih-alih menghabiskan waktu Anda menyediakan dan mengelola server. Latihan ini harus memakan waktu kurang dari 20 menit untuk diselesaikan, dan dimungkinkan dalam [Tingkat AWS Gratis](#).

Pertama, Anda membuat fungsi Lambda menggunakan konsol Lambda. Selanjutnya, Anda membuat REST API menggunakan konsol API Gateway REST API. Kemudian, Anda membuat metode API dan mengintegrasikannya dengan fungsi Lambda menggunakan integrasi proxy Lambda. Terakhir, Anda menerapkan dan menjalankan API Anda.

Saat Anda menjalankan REST API, API Gateway merutekan permintaan ke fungsi Lambda Anda. Lambda menjalankan fungsi dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons itu kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukan pengguna Akun AWS dan AWS Identity and Access Management (IAM) dengan akses konsol. Untuk informasi selengkapnya, lihat [Siapkan untuk menggunakan API Gateway](#).

Topik

- [Langkah 1: Membuat fungsi Lambda](#)
- [Langkah 2: Buat REST API](#)

- [Langkah 3: Buat integrasi proxy Lambda](#)
- [Langkah 4: Menerapkan API Anda](#)
- [Langkah 5: Panggil API Anda](#)
- [\(Opsional\) Langkah 6: Hapus](#)

Langkah 1: Membuat fungsi Lambda

Anda menggunakan fungsi Lambda untuk backend API Anda. Lambda menjalankan kode Anda hanya saat diperlukan dan menskalakan secara otomatis, dari beberapa permintaan per hari hingga ribuan per detik.

Untuk latihan ini, Anda menggunakan fungsi Node.js default di konsol Lambda.

Untuk membuat fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pilih Buat fungsi.
3. Di bagian Informasi dasar, untuk Nama fungsi, masukkan **my-function**.
4. Untuk semua opsi lain, gunakan pengaturan default.
5. Pilih Buat fungsi.

Kode fungsi Lambda default akan terlihat mirip dengan yang berikut ini:

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

Anda dapat memodifikasi fungsi Lambda untuk latihan ini, selama respons fungsi selaras dengan [format yang dibutuhkan API Gateway](#).

Ganti body respons default (Hello from Lambda!) dengan `The API Gateway REST API console is great!`. Ketika Anda memanggil fungsi contoh, ia mengembalikan 200 respons ke klien, bersama dengan respons yang diperbarui.

Langkah 2: Buat REST API

Selanjutnya, Anda membuat REST API dengan sumber daya root (/).

Untuk membuat REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Lakukan salah satu tindakan berikut:
 - Untuk membuat API pertama Anda, untuk REST API, pilih Build.
 - Jika Anda pernah membuat API sebelumnya, pilih Create API, lalu pilih Build for REST API.
3. Untuk nama API, masukkan **my-rest-api**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Langkah 3: Buat integrasi proxy Lambda

Selanjutnya, Anda membuat metode API untuk REST API di root resource (/) dan mengintegrasikan metode tersebut dengan fungsi Lambda Anda menggunakan integrasi proxy. Dalam integrasi proxy Lambda, API Gateway meneruskan permintaan masuk dari klien langsung ke fungsi Lambda.

Untuk membuat integrasi proxy Lambda

1. Pilih / sumber daya, lalu pilih Create method.
2. Untuk jenis Metode, pilih ANY.
3. Untuk jenis Integrasi, pilih Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, masukkan **my-function**, lalu pilih fungsi Lambda Anda.
6. Pilih metode Buat.

Langkah 4: Menerapkan API Anda

Selanjutnya, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **Prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Sekarang klien dapat menghubungi API Anda. Untuk menguji API Anda sebelum menerapkannya, Anda dapat memilih metode APAPUN secara opsional, menavigasi ke tab Uji, lalu pilih Uji.

Langkah 5: Panggil API Anda

Untuk menjalankan API Anda

1. Dari panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

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)

3. Masukkan URL pemanggilan di browser web.

URL lengkap akan terlihat seperti `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Browser Anda mengirimkan GET permintaan ke API.

4. Verifikasi respons API Anda. Anda akan melihat teks "The API Gateway REST API console is great!" di browser Anda.

(Opsional) Langkah 6: Hapus

Untuk mencegah biaya yang tidak perlu untuk Anda Akun AWS, hapus sumber daya yang Anda buat sebagai bagian dari latihan ini. Langkah-langkah berikut menghapus REST API Anda, fungsi Lambda Anda, dan sumber daya terkait.

Untuk menghapus REST API

1. Di panel Sumber Daya, pilih tindakan API, Hapus API.
2. Di kotak dialog Delete API, masukkan konfirmasi, lalu pilih Hapus.

Untuk menghapus fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda](https://console.aws.amazon.com/lambda).
2. Pada halaman Fungsi, pilih fungsi Anda. Pilih Tindakan, Hapus.
3. Di kotak dialog Hapus 1 fungsi, masukkan **delete**, lalu pilih Hapus.

Untuk menghapus grup log fungsi Lambda

1. Buka [Halaman grup log](#) di konsol Amazon CloudWatch.
2. Pada halaman Grup log, pilih grup log fungsi Anda (/aws/lambda/my-function). Kemudian, untuk Tindakan, pilih Hapus grup log.
3. Di kotak dialog Hapus grup log, pilih Hapus.

Untuk menghapus peran eksekusi fungsi Lambda

1. Buka [halaman Peran](#) dari konsol IAM.
2. (Opsional) Pada halaman Peran, di kotak pencarian, masukkan **my-function**.
3. Pilih peran fungsi Anda (misalnya, my-function-**31exxmpl**), lalu pilih Hapus.
4. Di Hapus **my-function-31exxmpl**? kotak dialog, masukkan nama peran, lalu pilih Hapus.

 Tip

Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan AWS CloudFormation atau AWS Serverless Application Model (AWS SAM). Untuk beberapa contoh CloudFormation template, lihat [contoh template untuk API Gateway](#) di repositori awsdocs GitHub .

Siapkan untuk menggunakan API Gateway

Sebelum Anda menggunakan Amazon API Gateway untuk pertama kalinya, selesaikan tugas-tugas berikut.

Mendaftar untuk Akun AWS

Untuk memulai AWS, Anda membutuhkan Akun AWS. Untuk informasi tentang membuat Akun AWS, lihat [Memulai dengan Akun AWS](#) di Panduan AWS Account Management Referensi.

Memulai dengan API Gateway

Dalam latihan memulai ini, Anda membuat API tanpa server. Tanpa server APIs memungkinkan Anda fokus pada aplikasi Anda, alih-alih menghabiskan waktu untuk menyediakan dan mengelola server. Latihan ini membutuhkan waktu kurang dari 20 menit untuk diselesaikan, dan dimungkinkan dalam [Tingkat AWS Gratis](#).

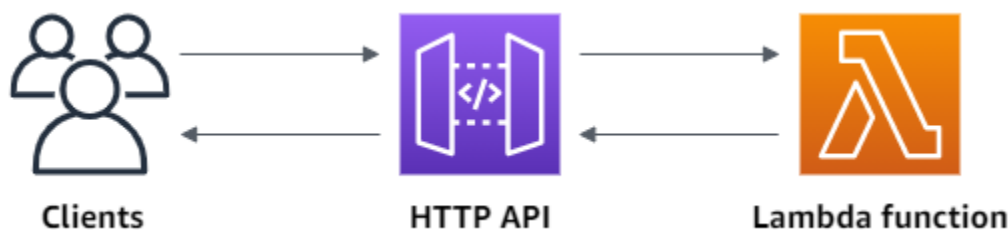
Pertama, Anda membuat fungsi Lambda menggunakan konsol. AWS Lambda Selanjutnya, Anda membuat API HTTP menggunakan konsol API Gateway. Kemudian, Anda menjalankan API Anda.

Note

Latihan ini menggunakan API HTTP. API Gateway juga mendukung REST APIs, yang mencakup lebih banyak fitur. Untuk tutorial menggunakan REST API, lihat [the section called “Memulai dengan konsol REST API”](#).

Untuk informasi selengkapnya tentang perbedaan antara HTTP APIs dan REST APIs, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

Saat Anda memanggil API HTTP, API Gateway merutekan permintaan ke fungsi Lambda Anda. Lambda menjalankan fungsi Lambda dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Topik

- [Langkah 1: Membuat fungsi Lambda](#)
- [Langkah 2: Buat API HTTP](#)
- [Langkah 3: Uji API Anda](#)
- [\(Opsional\) Langkah 4: Bersihkan](#)

- [Langkah selanjutnya](#)

Langkah 1: Membuat fungsi Lambda

Anda menggunakan fungsi Lambda untuk backend API Anda. Lambda menjalankan kode Anda hanya saat diperlukan dan menskalakan secara otomatis, dari beberapa permintaan per hari hingga ribuan per detik.

Untuk contoh ini, Anda menggunakan fungsi Node.js default dari konsol Lambda.

Untuk membuat fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pilih Buat fungsi.
3. Untuk Nama fungsi, masukkan **my-function**.
4. Untuk semua opsi lain, gunakan pengaturan default.
5. Pilih Buat fungsi.

Fungsi contoh mengembalikan 200 respon ke klien, dan teksHello from Lambda!.

Anda dapat memodifikasi fungsi Lambda, selama respons fungsi selaras dengan [format yang dibutuhkan API Gateway](#).

Kode fungsi Lambda default akan terlihat mirip dengan berikut ini:

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

Langkah 2: Buat API HTTP

Selanjutnya, Anda membuat API HTTP. API Gateway juga mendukung REST APIs dan WebSocket APIs, tetapi API HTTP adalah pilihan terbaik untuk latihan ini. REST APIs mendukung lebih banyak fitur daripada HTTP APIs, tetapi kami tidak memerlukan fitur tersebut untuk latihan ini. HTTP APIs

dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. WebSocket APIs pertahankan koneksi persisten dengan klien untuk komunikasi dupleks penuh, yang tidak diperlukan untuk contoh ini.

HTTP API menyediakan endpoint HTTP untuk fungsi Lambda Anda. API Gateway merutekan permintaan ke fungsi Lambda Anda, lalu mengembalikan respons fungsi ke klien.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Lakukan salah satu hal berikut ini:
 - Untuk membuat API pertama Anda, untuk HTTP API, pilih Build.
 - Jika Anda telah membuat API sebelumnya, pilih Buat API, lalu pilih Build for HTTP API.
3. Untuk Integrasi, pilih Tambahkan integrasi.
4. Pilih Lambda.
5. Untuk fungsi Lambda, masukkan **my-function**
6. Untuk nama API, masukkan **my-http-api**.
7. Pilih Berikutnya.
8. Tinjau rute yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
9. Tinjau tahap yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
10. Pilih Buat.

Sekarang Anda telah membuat API HTTP dengan integrasi Lambda yang siap menerima permintaan dari klien.

Langkah 3: Uji API Anda

Selanjutnya, Anda menguji API Anda untuk memastikan bahwa itu berfungsi. Untuk mempermudah, gunakan browser web untuk menjalankan API Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Perhatikan URL pemanggilan API Anda.

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

- Salin URL pemanggilan API Anda, dan masukkan di browser web. Tambahkan nama fungsi Lambda Anda ke URL pemanggilan Anda untuk memanggil fungsi Lambda Anda. Secara default, konsol API Gateway membuat rute dengan nama yang sama dengan fungsi Lambda Anda. `my-function`

URL lengkap akan terlihat seperti `https://abcdef123.execute-api.us-east-2.amazonaws.com/my-function`.

Browser Anda mengirimkan GET permintaan ke API.

- Verifikasi respons API Anda. Anda akan melihat teks "Hello from Lambda!" di browser Anda.

(Opsional) Langkah 4: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari latihan memulai ini. Langkah-langkah berikut menghapus API HTTP Anda, fungsi Lambda Anda, dan sumber daya terkait.

Untuk menghapus API HTTP

- Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
- Pada APIshalaman, pilih API. Pilih Tindakan, lalu pilih Hapus.
- Pilih Hapus.

Untuk menghapus fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pada halaman Fungsi, pilih fungsi. Pilih Tindakan, lalu pilih Hapus.
3. Pilih Hapus.

Untuk menghapus grup log fungsi Lambda

1. Di CloudWatch konsol Amazon, buka [halaman Grup log](#).
2. Pada halaman Grup log, pilih grup log fungsi (/aws/lambda/my-function). Pilih Tindakan, lalu pilih Hapus grup log.
3. Pilih Hapus.

Untuk menghapus peran eksekusi fungsi Lambda

1. Di AWS Identity and Access Management konsol, buka [halaman Peran](#).
2. Pilih peran fungsi, misalnya, my-function-*31exxmpl*.
3. Pilih Hapus peran.
4. Pilih Ya, Hapus.

Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan CloudFormation atau AWS SAM Misalnya CloudFormation template, lihat [contoh CloudFormation template](#).

Langkah selanjutnya

Untuk contoh ini, Anda menggunakan Konsol Manajemen AWS untuk membuat HTTP API sederhana. HTTP API memanggil fungsi Lambda dan mengembalikan respons ke klien.

Berikut ini adalah langkah selanjutnya saat Anda terus bekerja dengan API Gateway.

- [Konfigurasi jenis integrasi API tambahan](#), termasuk:
 - [Titik akhir HTTP](#)
 - [Sumber daya pribadi dalam VPC, seperti layanan Amazon ECS](#)

- [AWS layanan seperti Amazon Simple Queue Service AWS Step Functions, dan Kinesis Data Streams](#)
- [Kontrol akses ke Anda APIs](#)
- [Aktifkan pencatatan untuk Anda APIs](#)
- [Konfigurasi throttling untuk Anda APIs](#)
- [Konfigurasi domain khusus untuk Anda APIs](#)

Untuk mendapatkan bantuan terkait Amazon API Gateway dari komunitas, lihat [Forum Diskusi API Gateway](#). Ketika Anda memasuki forum ini, AWS mungkin mengharuskan Anda untuk masuk.

Untuk mendapatkan bantuan dengan API Gateway langsung dari AWS, lihat opsi dukungan di [halaman AWS Support](#).

Lihat juga [pertanyaan yang sering diajukan \(FAQs\)](#), atau [hubungi kami secara langsung](#).

Tutorial dan lokakarya Amazon API Gateway

Tutorial dan lokakarya berikut menyediakan latihan langsung untuk membantu Anda mempelajari API Gateway. Untuk memulai, kami sarankan Anda menyelesaikannya [the section called “Tutorial: Buat REST API dengan integrasi proxy Lambda”](#).

Tutorial REST API

- [Pilih tutorial AWS Lambda integrasi](#)
- [Tutorial: Buat REST API dengan mengimpor contoh](#)
- [Pilih tutorial integrasi HTTP](#)
- [Tutorial: Buat REST API dengan integrasi pribadi](#)
- [Tutorial: Buat REST API dengan AWS integrasi](#)
- [Tutorial: Buat REST API kalkulator dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda](#)
- [Tutorial: Buat REST API sebagai proxy Amazon S3](#)
- [Tutorial: Buat REST API sebagai proxy Amazon Kinesis](#)
- [Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI](#)
- [Tutorial: Membuat REST API pribadi](#)

HTTP API tutorial

- [Tutorial: Buat API HTTP CRUD dengan Lambda dan DynamoDB](#)
- [Tutorial: Buat API HTTP dengan integrasi pribadi ke layanan Amazon ECS](#)

WebSocket API tutorial

- [Tutorial: Buat aplikasi WebSocket obrolan dengan WebSocket API, Lambda, dan DynamoDB](#)

Lokakarya

- [Membangun aplikasi web tanpa server](#)
- [CI/CD untuk aplikasi tanpa server](#)
- [Bengkel keamanan tanpa server](#)

- [Manajemen identitas tanpa server, otentikasi dan otorisasi](#)
- [Lokakarya Amazon API Gateway](#)

Tutorial API REST API Amazon API Gateway

Tutorial berikut menyediakan latihan langsung untuk membantu Anda mempelajari API Gateway REST. APIs

Topik

- [Pilih tutorial AWS Lambda integrasi](#)
- [Tutorial: Buat REST API dengan mengimpor contoh](#)
- [Pilih tutorial integrasi HTTP](#)
- [Tutorial: Buat REST API dengan integrasi pribadi](#)
- [Tutorial: Buat REST API dengan AWS integrasi](#)
- [Tutorial: Buat REST API kalkulator dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda](#)
- [Tutorial: Buat REST API sebagai proxy Amazon S3](#)
- [Tutorial: Buat REST API sebagai proxy Amazon Kinesis](#)
- [Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI](#)
- [Tutorial: Membuat REST API pribadi](#)

Pilih tutorial AWS Lambda integrasi

Untuk membangun API dengan integrasi Lambda, Anda dapat menggunakan integrasi proxy Lambda atau integrasi non-proxy Lambda.

Dalam integrasi proxy Lambda, input ke fungsi Lambda dapat dinyatakan sebagai kombinasi header permintaan, variabel jalur, parameter string kueri, isi, dan data konfigurasi API. Anda hanya perlu memilih fungsi Lambda. API Gateway mengonfigurasi permintaan integrasi dan respons integrasi untuk Anda. Setelah disiapkan, metode API Anda dapat berkembang tanpa mengubah pengaturan yang ada. Ini dimungkinkan karena fungsi backend Lambda mem-parsing data permintaan yang masuk dan merespons klien.

Dalam integrasi non-proxy Lambda, Anda harus memastikan bahwa input ke fungsi Lambda disediakan sebagai payload permintaan integrasi. Anda harus memetakan data input apa pun yang

diberikan klien sebagai parameter permintaan ke dalam badan permintaan integrasi yang tepat. Anda mungkin juga perlu menerjemahkan badan permintaan yang disediakan klien ke dalam format yang dikenali oleh fungsi Lambda.

Baik dalam proxy Lambda atau integrasi non-proxy Lambda, Anda dapat menggunakan fungsi Lambda di akun yang berbeda dari akun tempat Anda membuat API.

Topik

- [Tutorial: Buat REST API dengan integrasi proxy Lambda](#)
- [Tutorial: Buat REST API dengan integrasi non-proxy Lambda](#)
- [Tutorial: Buat REST API dengan integrasi proxy Lambda lintas akun](#)

Tutorial: Buat REST API dengan integrasi proxy Lambda

[Integrasi proxy Lambda adalah jenis integrasi](#) API Gateway API yang ringan dan fleksibel yang memungkinkan Anda mengintegrasikan metode API — atau seluruh API — dengan fungsi Lambda. Fungsi Lambda dapat ditulis dalam [bahasa apa pun yang didukung Lambda](#). Karena ini adalah integrasi proxy, Anda dapat mengubah implementasi fungsi Lambda kapan saja tanpa perlu menerapkan ulang API Anda.

Dalam tutorial ini, Anda akan melakukan hal-hal berikut:

- Buat “Halo, Dunia!” Lambda berfungsi sebagai backend untuk API.
- Buat dan uji “Halo, Dunia!” API dengan integrasi proxy Lambda.

Topik

- [Buat “Halo, Dunia!” Fungsi Lambda](#)
- [Buat “Halo, Dunia!” API](#)
- [Menerapkan dan menguji API](#)

Buat “Halo, Dunia!” Fungsi Lambda

Untuk membuat “Halo, Dunia!” Fungsi Lambda di konsol Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pada bilah AWS navigasi, pilih file [Wilayah AWS](#).

 Note

Perhatikan wilayah tempat Anda membuat fungsi Lambda. Anda akan membutuhkannya saat membuat API.

3. Pilih Fungsi di panel navigasi.
4. Pilih Buat fungsi.
5. Pilih Penulis dari scratch.
6. Di bagian Informasi dasar, lakukan hal berikut:
 - a. Dalam nama Fungsi, masukkan **GetStartedLambdaProxyIntegration**.
 - b. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
 - c. Untuk Arsitektur, pertahankan pengaturan default.
 - d. (Opsional) Di bagian Izin, luaskan Ubah peran eksekusi default. Untuk daftar dropdown peran eksekusi, pilih Buat peran baru dari templat AWS kebijakan.
 - e. Dalam nama Peran, masukkan **GetStartedLambdaBasicExecutionRole**.
 - f. Biarkan bidang Policy templates kosong.
 - g. Pilih Buat fungsi.
7. Di bawah Kode fungsi, dalam editor kode inline, kode copy/paste berikut:

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;
    }
  }
  return res;
```

```

    }
    } 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. Pilih Deploy.

Buat “Halo, Dunia!” API

Sekarang buat API untuk “Hello, World!” Lambda berfungsi dengan menggunakan konsol API Gateway.

Untuk membuat “Halo, Dunia!” API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **LambdaProxyAPI**.

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk contoh ini, Anda membuat sumber daya / helloworld.

Untuk membuat sumber daya

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai /.
4. Untuk Nama sumber daya, masukkan **helloworld**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Dalam integrasi proxy, seluruh permintaan dikirim ke fungsi Lambda backend apa adanya, melalui metode catch-all ANY yang mewakili metode HTTP apa pun. Metode HTTP sebenarnya ditentukan oleh klien pada waktu berjalan. ANYMetode ini memungkinkan Anda untuk menggunakan penyiapan metode API tunggal untuk semua metode HTTP yang didukung:DELETE,GET,HEAD,OPTIONS,PATCH,POST, danPUT.

Untuk membuat **ANY** metode

1. Pilih sumber daya /helloworld, lalu pilih Create method.
2. Untuk jenis Metode, pilih APAPUN.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda, lalu masukkan nama fungsi.

6. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000
7. Pilih metode Buat.

Menerapkan dan menguji API

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **test**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.
6. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Gunakan browser dan cURL untuk menguji API dengan integrasi proxy Lambda

Anda dapat menggunakan browser atau [cURL](#) untuk menguji API Anda.

Untuk menguji GET permintaan hanya menggunakan parameter string kueri, Anda dapat memasukkan URL untuk helloworld sumber daya API ke dalam bilah alamat browser.

Untuk membuat URL sumber daya API, tambahkan helloworld sumber daya helloworld dan parameter ?greeter=John string kueri ke URL pemanggilan Anda. URL Anda akan terlihat seperti berikut ini.

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

Untuk metode lain, Anda harus menggunakan utilitas pengujian REST API yang lebih canggih, seperti [POSTMAN](#) atau [cURL](#). Tutorial ini menggunakan cURL. Contoh perintah cURL di bawah ini mengasumsikan bahwa cURL diinstal pada komputer Anda.

Untuk menguji API yang Anda gunakan menggunakan cURL:

1. Buka jendela terminal.

2. Salin perintah cURL berikut dan tempel ke jendela terminal, dan ganti URL pemanggilan dengan yang Anda salin di langkah sebelumnya dan tambahkan **/helloworld** ke akhir URL.

 Note

Jika Anda menjalankan perintah di Windows, gunakan sintaks ini sebagai gantinya:

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

- a. Untuk memanggil API dengan parameter string kueri?greeter=John:

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

- b. Untuk memanggil API dengan parameter headergreeter: John:

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

- c. Untuk memanggil API dengan badan{"greeter": "John"}:

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

Dalam semua kasus, outputnya adalah respons 200 dengan badan respons berikut:

```
Hello, John!
```

Tutorial: Buat REST API dengan integrasi non-proxy Lambda

Dalam panduan ini, kami menggunakan konsol API Gateway untuk membangun API yang memungkinkan klien memanggil fungsi Lambda melalui integrasi non-proxy Lambda (juga dikenal sebagai integrasi khusus). Untuk informasi selengkapnya tentang AWS Lambda dan fungsi Lambda, lihat Panduan [AWS Lambda Pengembang](#).

Untuk memfasilitasi pembelajaran, kami memilih fungsi Lambda sederhana dengan pengaturan API minimal untuk memandu Anda melalui langkah-langkah membangun API Gateway API dengan integrasi kustom Lambda. Bila perlu, kami menjelaskan beberapa logika. Untuk contoh lebih rinci tentang integrasi kustom Lambda, lihat. [Tutorial: Buat REST API kalkulator dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda](#)

Sebelum membuat API, siapkan backend Lambda dengan membuat fungsi Lambda di, dijelaskan selanjutnya. AWS Lambda

Topik

- [Buat fungsi Lambda untuk integrasi non-proxy Lambda](#)
- [Buat API dengan integrasi non-proxy Lambda](#)
- [Uji pemanggilan metode API](#)
- [Terapkan API](#)
- [Uji API dalam tahap penerapan](#)
- [Bersihkan](#)

Buat fungsi Lambda untuk integrasi non-proxy Lambda

Note

Membuat fungsi Lambda dapat mengakibatkan biaya ke akun Anda AWS .

Pada langkah ini, Anda membuat “Halo, Dunia!” -seperti fungsi Lambda untuk integrasi kustom Lambda. Sepanjang panduan ini, fungsi ini disebut. `GetStartedLambdaIntegration`

Implementasi fungsi `GetStartedLambdaIntegration` Lambda ini adalah sebagai berikut:

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}
```

Untuk integrasi kustom Lambda, API Gateway meneruskan input ke fungsi Lambda dari klien sebagai badan permintaan integrasi. `eventObjek` dari fungsi handler Lambda adalah input.

Fungsi Lambda kami sederhana. Ini mem-parsing event objek input untuk `name`, `city`, `time`, dan `day` properti. Kemudian mengembalikan salam, sebagai objek JSON dari `{"message":greeting}`, ke penelepon. Pesannya ada dalam "Good [morning|afternoon|day], [*name*|you] in [*city*|World]. Happy *day*!" pola. Diasumsikan bahwa input ke fungsi Lambda adalah dari objek JSON berikut:

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

```
}
```

Lihat informasi selengkapnya di [Panduan Developer AWS Lambda](#).

Selain itu, fungsi mencatat eksekusinya ke Amazon CloudWatch dengan `menelepon console.log(...)`. Ini berguna untuk melacak panggilan saat men-debug fungsi. Untuk mengizinkan `GetStartedLambdaIntegration` fungsi mencatat panggilan, tetapkan peran IAM dengan kebijakan yang sesuai untuk fungsi Lambda untuk membuat CloudWatch aliran dan menambahkan entri log ke aliran. Konsol Lambda memandu Anda untuk membuat peran dan kebijakan IAM yang diperlukan.

Jika Anda menyiapkan API tanpa menggunakan konsol API Gateway, seperti saat [mengimpor API dari file `OpenAPI`](#), Anda harus secara eksplisit membuat, jika perlu, dan menyiapkan peran dan kebijakan pemanggilan untuk API Gateway untuk menjalankan fungsi Lambda. Untuk informasi selengkapnya tentang cara menyiapkan peran pemanggilan dan eksekusi Lambda untuk API Gateway API, lihat. [Kontrol akses ke REST API dengan izin IAM](#)

Dibandingkan dengan `GetStartedLambdaProxyIntegration`, fungsi Lambda untuk integrasi proxy Lambda, fungsi Lambda untuk integrasi kustom `GetStartedLambdaIntegration` Lambda hanya mengambil masukan dari badan permintaan integrasi API Gateway API. Fungsi ini dapat mengembalikan output dari setiap objek JSON, string, angka, Boolean, atau bahkan blob biner. Fungsi Lambda untuk integrasi proxy Lambda, sebaliknya, dapat mengambil input dari data permintaan apa pun, tetapi harus mengembalikan output dari objek JSON tertentu. `GetStartedLambdaIntegration` Fungsi untuk integrasi kustom Lambda dapat memiliki parameter permintaan API sebagai input, asalkan API Gateway memetakan parameter permintaan API yang diperlukan ke badan permintaan integrasi sebelum meneruskan permintaan klien ke backend. Agar hal ini terjadi, pengembang API harus membuat template pemetaan dan mengonfigurasinya pada metode API saat membuat API.

Sekarang, buat fungsi `GetStartedLambdaIntegration` Lambda.

Untuk membuat fungsi **`GetStartedLambdaIntegration`** Lambda untuk integrasi kustom Lambda

1. Buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
2. Lakukan salah satu tindakan berikut:
 - Jika halaman selamat datang muncul, pilih Mulai Sekarang dan kemudian pilih Buat fungsi.
 - Jika halaman daftar Lambda > Fungsi muncul, pilih Buat fungsi.
3. Pilih Tulis dari awal.

4. Di panel Penulis dari awal, lakukan hal berikut:
 - a. Untuk Nama, masukkan **GetStartedLambdaIntegration** sebagai nama fungsi Lambda.
 - b. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
 - c. Untuk Arsitektur, pertahankan pengaturan default.
 - d. (Opsional) Di bagian Izin, luaskan Ubah peran eksekusi default. Untuk daftar dropdown peran eksekusi, pilih Buat peran baru dari templat AWS kebijakan.
 - e. Untuk nama Peran, masukkan nama untuk peran Anda (misalnya, **GetStartedLambdaIntegrationRole**).
 - f. Untuk templat Kebijakan, pilih Izin microservice sederhana.
 - g. Pilih Buat fungsi.
5. Di panel Configure function, di bawah Kode fungsi lakukan hal berikut:
 - a. Salin kode fungsi Lambda yang tercantum di awal bagian ini dan tempel di editor kode sebaris.
 - b. Tinggalkan pilihan default untuk semua bidang lain di bagian ini.
 - c. Pilih Deploy.
6. Untuk menguji fungsi yang baru dibuat, pilih tab Uji.
 - a. Untuk Nama peristiwa, masukkan **HelloWorldTest**.
 - b. Untuk Event JSON, ganti kode default dengan yang berikut ini.

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

- c. Pilih Uji untuk menjalankan fungsi. Hasil eksekusi: bagian berhasil ditampilkan. Perluas Detail dan Anda melihat output berikut.

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

Outputnya juga ditulis ke CloudWatch Log.

Sebagai latihan sampingan, Anda dapat menggunakan konsol IAM untuk melihat peran IAM (GetStartedLambdaIntegrationRole) yang dibuat sebagai bagian dari pembuatan fungsi Lambda. Terlampir pada peran IAM ini adalah dua kebijakan inline. Satu menetapkan izin paling dasar untuk eksekusi Lambda. Ini memungkinkan memanggil CloudWatch sumber daya apa pun dari akun Anda di wilayah tempat fungsi Lambda dibuat. CloudWatch CreateLogGroup Kebijakan ini juga memungkinkan pembuatan CloudWatch stream dan peristiwa logging untuk fungsi GetStartedLambdaIntegration Lambda.

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:*"
      ]
    }
  ]
}
```

Dokumen kebijakan lainnya berlaku untuk memanggil AWS layanan lain yang tidak digunakan dalam contoh ini. Anda dapat melewatkannya untuk saat ini.

Terkait dengan peran IAM adalah entitas tepercaya, yaitu `lambda.amazonaws.com`. Inilah hubungan kepercayaannya:

JSON

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

Kombinasi hubungan kepercayaan ini dan kebijakan sebaris memungkinkan fungsi Lambda memanggil fungsi untuk mencatat peristiwa ke `console.log()` Log. CloudWatch

Buat API dengan integrasi non-proxy Lambda

Dengan fungsi Lambda (`GetStartedLambdaIntegration`) yang dibuat dan diuji, Anda siap mengekspos fungsi tersebut melalui API Gateway API. Untuk tujuan ilustrasi, kami mengekspos fungsi Lambda dengan metode HTTP generik. Kami menggunakan badan permintaan, variabel jalur URL, string kueri, dan header untuk menerima data input yang diperlukan dari klien. Kami mengaktifkan validator permintaan API Gateway untuk API untuk memastikan bahwa semua data yang diperlukan didefinisikan dan ditentukan dengan benar. Kami mengonfigurasi template pemetaan untuk API Gateway untuk mengubah data permintaan yang disediakan klien menjadi format yang valid seperti yang dipersyaratkan oleh fungsi Lambda backend.

Untuk membuat API dengan integrasi non-proxy Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **LambdaNonProxyAPI**.

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Setelah membuat API Anda, Anda membuat sumber daya/{city}. Ini adalah contoh sumber daya dengan variabel jalur yang mengambil masukan dari klien. Kemudian, Anda memetakan variabel jalur ini ke input fungsi Lambda menggunakan template pemetaan.

Untuk membuat sumber daya

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai /.
4. Untuk Nama sumber daya, masukkan **{city}**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Setelah membuat sumber daya/{city} Anda, Anda membuat ANY metode. Kata kerja ANY HTTP adalah placeholder untuk metode HTTP yang valid yang klien kirimkan pada waktu berjalan. Contoh ini menunjukkan bahwa ANY metode dapat digunakan untuk integrasi kustom Lambda serta untuk integrasi proxy Lambda.

Untuk membuat **ANY** metode

1. Pilih sumber daya/{city}, lalu pilih Create method.
2. Untuk jenis Metode, pilih APAPUN.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Matikan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda, lalu masukkan nama fungsi.
6. Pilih Pengaturan permintaan Metode.

Sekarang, Anda mengaktifkan validator permintaan untuk variabel jalur URL, parameter string kueri, dan header untuk memastikan bahwa semua data yang diperlukan ditentukan. Untuk contoh ini, Anda membuat parameter string `time` kueri dan `day` header.

7. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header.
8. Pilih parameter string kueri URL dan lakukan hal berikut:
 - a. Pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan **`time`**.
 - c. Aktifkan Diperlukan.
 - d. Tetap caching dimatikan.
9. Pilih header permintaan HTTP dan lakukan hal berikut:
 - a. Pilih Tambahkan header.
 - b. Untuk Nama, masukkan **`day`**.
 - c. Aktifkan Diperlukan.
 - d. Tetap caching dimatikan.
10. Pilih metode Buat.

Setelah mengaktifkan validator permintaan, Anda mengonfigurasi permintaan integrasi untuk ANY metode dengan menambahkan templat pemetaan tubuh untuk mengubah permintaan yang masuk menjadi payload JSON, seperti yang dipersyaratkan oleh fungsi Lambda backend.

Untuk mengkonfigurasi permintaan integrasi

1. Pada tab Permintaan integrasi, di bawah pengaturan permintaan Integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih Template pemetaan.
4. Pilih Tambahkan templat pemetaan.
5. Untuk jenis Konten, masukkan **`application/json`**.
6. Untuk badan Template, masukkan kode berikut:

```
#set($inputRoot = $input.path('$'))
{
  "city": "$input.params('city')",
```

```
"time": "$input.params('time')",  
"day": "$input.params('day')",  
"name": "$inputRoot.callerName"  
}
```

7. Pilih Simpan.

Uji pemanggilan metode API

Konsol API Gateway menyediakan fasilitas pengujian bagi Anda untuk menguji pemanggilan API sebelum diterapkan. Anda menggunakan fitur Uji konsol untuk menguji API dengan mengirimkan permintaan berikut:

```
POST /Seattle?time=morning  
day:Wednesday  
  
{  
  "callerName": "John"  
}
```

Dalam permintaan pengujian ini, Anda akan mengatur ANY kePOST, mengatur {city} keSeattle, menetapkan Wednesday sebagai nilai day header, dan menetapkan "John" sebagai nilai callerName.

Untuk menguji **ANY** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk jenis Metode, pilih **POST**.
3. Untuk Path, di bawah kota, masuk **Seattle**.
4. Untuk string Query, masukkan **time=morning**.
5. Untuk Header, masukkan **day:Wednesday**.
6. Untuk Badan Permintaan, masukkan **{ "callerName": "John" }**.
7. Pilih Uji.

Verifikasi bahwa muatan respons yang dikembalikan adalah sebagai berikut:

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



```
}

```

Anda juga dapat melihat log untuk memeriksa bagaimana API Gateway memproses permintaan dan respons.

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/YUnT2Rau0L9HCG5p7FC05h3Ivw1FfvcidQNXeYvsKJTLXI05/yEnY3ttIANpNYL0ezD9Es8rBfyruHfJf0qextKlsC8DymCcqlGkig8qLKcZ0hWJWwiPJiFgL7laabXs++ZhCa4hdZo4iq1G729DE4gaV1mJVdoAagIUwLMO+y4NxFDu0r7I0/E05nYcCrrpGVVBYiGk7H4T6sXuhTkbNNqVmXtV3ch5b0lh7 [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
```

Log menampilkan permintaan masuk sebelum pemetaan dan permintaan integrasi setelah pemetaan. Ketika pengujian gagal, log berguna untuk mengevaluasi apakah input asli benar atau template pemetaan berfungsi dengan benar.

Terapkan API

Doa tes adalah simulasi dan memiliki keterbatasan. Misalnya, ia melewati mekanisme otorisasi apa pun yang diberlakukan pada API. Untuk menguji eksekusi API secara real time, Anda harus menerapkan API terlebih dahulu. Untuk menerapkan API, Anda membuat tahapan untuk membuat snapshot API pada saat itu. Nama stage juga mendefinisikan path dasar setelah nama host default API. Sumber daya root API ditambahkan setelah nama panggung. Saat memodifikasi API, Anda harus menerapkannya kembali ke tahap baru atau yang sudah ada sebelum perubahan diterapkan.

Untuk menerapkan API ke panggung

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **test**.

Note

Input harus berupa teks yang dikodekan UTF-8 (yaitu, tidak terlokalisasi).

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda. Pola umum URL dasar API adalah `https://api-id.region.amazonaws.com/stageName`. Misalnya, URL dasar API (beags1mnid) yang dibuat di us-west-2 wilayah dan diterapkan ke test panggung adalah `https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Uji API dalam tahap penerapan

Ada beberapa cara Anda dapat menguji API yang diterapkan. Untuk permintaan GET yang hanya menggunakan variabel jalur URL atau parameter string kueri, Anda dapat memasukkan URL sumber daya API di browser. Untuk metode lain, Anda harus menggunakan utilitas pengujian REST API yang lebih canggih, seperti [POSTMAN](#) atau [cURL](#).

Untuk menguji API menggunakan cURL

1. Buka jendela terminal di komputer lokal Anda yang terhubung ke internet.
2. Untuk menguji POST /Seattle?time=evening:

Salin perintah cURL berikut dan tempel ke jendela terminal.

```
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"
}'
```

Anda harus mendapatkan respons yang sukses dengan muatan berikut:

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

Jika Anda POST mengubah PUT permintaan metode ini, Anda mendapatkan respons yang sama.

Bersihkan

Jika Anda tidak lagi membutuhkan fungsi Lambda yang Anda buat untuk panduan ini, Anda dapat menghapusnya sekarang. Anda juga dapat menghapus sumber daya IAM yang menyertainya.

 Warning

Jika Anda berencana untuk menyelesaikan penelusuran lain dalam seri ini, jangan hapus peran eksekusi Lambda atau peran pemanggilan Lambda. Jika Anda menghapus fungsi Lambda yang Anda APIs andalkan, fungsi tersebut tidak APIs akan berfungsi lagi. Menghapus fungsi Lambda tidak dapat dibatalkan. Jika Anda ingin menggunakan fungsi Lambda lagi, Anda harus membuat ulang fungsi tersebut.

Jika Anda menghapus sumber daya IAM yang bergantung pada fungsi Lambda, fungsi Lambda itu tidak akan berfungsi lagi, dan APIs semua yang bergantung pada fungsi itu tidak akan berfungsi lagi. Menghapus sumber daya IAM tidak dapat dibatalkan. Jika Anda ingin menggunakan sumber daya IAM lagi, Anda harus membuat ulang sumber daya.

Untuk menghapus fungsi Lambda

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
2. Dari daftar fungsi, pilih `GetStartedLambdaIntegration`, pilih Tindakan, lalu pilih Hapus fungsi. Saat diminta, pilih Hapus lagi.

Untuk menghapus sumber daya IAM terkait

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Dari Detail, pilih Peran.
3. Dari daftar peran, pilih `GetStartedLambdaIntegrationRole`, pilih Tindakan Peran, lalu pilih Hapus Peran. Ikuti langkah-langkah di konsol untuk menghapus peran.

Tutorial: Buat REST API dengan integrasi proxy Lambda lintas akun

Anda sekarang dapat menggunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai backend integrasi API Anda. Setiap akun dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Ini membuatnya mudah untuk mengelola dan berbagi fungsi backend Lambda secara terpusat di beberapa APIs

Di bagian ini, kami menunjukkan cara mengonfigurasi integrasi proxy Lambda lintas akun menggunakan konsol Amazon API Gateway.

Buat API untuk integrasi Lambda lintas akun API Gateway

Untuk membuat API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **CrossAccountLambdaAPI**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Buat fungsi integrasi Lambda di akun lain

Sekarang Anda akan membuat fungsi Lambda di akun yang berbeda dari yang Anda buat contoh API.

Membuat fungsi Lambda di akun lain

1. Masuk ke konsol Lambda di akun yang berbeda dari akun tempat Anda membuat API Gateway API.
2. Pilih Buat fungsi.
3. Pilih Penulis dari scratch.
4. Di bawah Tulis dari awal, lakukan langkah berikut:
 - a. Untuk nama Fungsi, masukkan nama.
 - b. Dari daftar drop-down Runtime, pilih runtime Node.js yang didukung.
 - c. Untuk Arsitektur, pertahankan pengaturan default.
 - d. Di bagian Izin, perluas Pilih atau buat peran eksekusi. Anda dapat membuat peran atau memilih peran yang ada.
 - e. Pilih fungsi Buat untuk melanjutkan.

5. Gulir ke bawah ke panel kode Fungsi.
6. Masukkan implementasi fungsi Node.js dari [the section called “Tutorial: Buat REST API dengan integrasi proxy Lambda”](#).
7. Pilih Deploy.
8. Perhatikan ARN lengkap untuk fungsi Anda (di sudut kanan atas panel fungsi Lambda). Anda akan membutuhkannya saat membuat integrasi Lambda lintas akun Anda.

Konfigurasi integrasi Lambda lintas akun

Setelah Anda memiliki fungsi integrasi Lambda di akun yang berbeda, Anda dapat menggunakan konsol API Gateway untuk menambahkannya ke API Anda di akun pertama Anda.

Note

Jika Anda mengonfigurasi otorisasi lintas wilayah dan lintas akun, `sourceArn` yang ditambahkan ke fungsi target harus menggunakan wilayah fungsi, bukan wilayah API.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk contoh ini, Anda membuat sumber daya / helloworld.

Untuk membuat sumber daya

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai /.
4. Untuk Nama sumber daya, masukkan **helloworld**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Setelah Anda membuat sumber daya, Anda membuat GET metode. Anda mengintegrasikan GET metode dengan fungsi Lambda di akun lain.

Untuk membuat **GET** metode

1. Pilih sumber daya /helloworld, lalu pilih Create method.

2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, masukkan ARN lengkap fungsi Lambda Anda dari Langkah 1.

Di konsol Lambda, Anda dapat menemukan ARN untuk fungsi Anda di sudut kanan atas jendela konsol.

6. Saat Anda memasukkan ARN, string `aws lambda add-permission` perintah akan muncul. Kebijakan ini memberikan akses akun pertama Anda ke fungsi Lambda akun kedua Anda. Salin dan tempel string `aws lambda add-permission` perintah ke AWS CLI jendela yang dikonfigurasi untuk akun kedua Anda.
7. Pilih metode Buat.

Anda dapat melihat kebijakan terbaru untuk fungsi Anda di konsol Lambda.

(Opsional) Untuk melihat kebijakan Anda yang diperbarui

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di <https://console.aws.amazon.com/lambda/>.
2. Pilih fungsi Lambda Anda.
3. Pilih Izin.

Anda akan melihat Allow kebijakan dengan Condition klausa di mana dalam `AWS:SourceArn` adalah ARN untuk metode API GET Anda.

Tutorial: Buat REST API dengan mengimpor contoh

Anda dapat menggunakan konsol Amazon API Gateway untuk membuat dan menguji REST API sederhana dengan integrasi HTTP untuk PetStore situs web. Definisi API telah dikonfigurasi sebelumnya sebagai file OpenAPI 2.0. Setelah memuat definisi API ke dalam API Gateway, Anda dapat menggunakan konsol API Gateway untuk memeriksa struktur dasar API atau cukup menerapkan dan menguji API.

PetStore Contoh API mendukung metode berikut bagi klien untuk mengakses situs backend HTTP dari `http://petstore-demo-endpoint.execute-api.com/petstore/pets`

 Note

Tutorial ini menggunakan endpoint HTTP sebagai contoh. Saat Anda membuat sendiri APIs, kami sarankan Anda menggunakan titik akhir HTTPS untuk integrasi HTTP Anda.

- GET `/`: untuk akses baca sumber daya root API yang tidak terintegrasi dengan titik akhir backend apa pun. API Gateway merespons dengan ikhtisar PetStore situs web. Ini adalah contoh dari jenis MOCK integrasi.
- GET `/pets`: untuk akses baca ke sumber daya API yang terintegrasi dengan `/pets` sumber daya backend `/pets` bernama sama. Backend mengembalikan halaman hewan peliharaan yang tersedia di file. PetStore Ini adalah contoh dari jenis HTTP integrasi. URL dari titik akhir integrasi adalah `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- POST `/pets`: untuk akses tulis ke `/pets` sumber daya API yang terintegrasi dengan sumber daya backend `/petstore/pets`. Setelah menerima permintaan yang benar, backend menambahkan hewan peliharaan yang ditentukan ke PetStore dan mengembalikan hasilnya ke pemanggil. Integrasi juga HTTP.
- GET `/pets/{petId}`: untuk akses baca ke hewan peliharaan yang diidentifikasi oleh `petId` nilai sebagaimana ditentukan sebagai variabel jalur dari URL permintaan masuk. Metode ini juga memiliki tipe HTTP integrasi. Backend mengembalikan hewan peliharaan tertentu yang ditemukan di file. PetStore URL titik akhir HTTP backend adalah `http://petstore-demo-endpoint.execute-api.com/petstore/pets/n`, di mana *n* bilangan bulat sebagai pengidentifikasi hewan peliharaan yang ditanyakan.

API mendukung akses CORS melalui OPTIONS metode jenis MOCK integrasi. API Gateway mengembalikan header yang diperlukan yang mendukung akses CORS.

Prosedur berikut memandu Anda melalui langkah-langkah untuk membuat dan menguji API dari contoh menggunakan API Gateway Console.

Untuk mengimpor, membangun, dan menguji contoh API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Lakukan salah satu tindakan berikut:
 - Untuk membuat API pertama Anda, untuk REST API, pilih Build.
 - Jika Anda telah membuat API sebelumnya, pilih Buat API, lalu pilih Build for REST API.

- Di bawah Create REST API, pilih Example API lalu pilih Create API untuk membuat contoh API.

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        ],
```

Anda dapat menggulir ke bawah definisi OpenAPI untuk detail contoh API ini sebelum memilih Create API.

- Di panel navigasi utama, pilih Resources. API yang baru dibuat ditampilkan sebagai berikut:

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

Resources

API actions ▼

Deploy API

Create resource

☐ /

GET

☐ /pets

GET

OPTIONS

POST

☐ /{petId}

GET

OPTIONS

Resource details

Update documentation

Enable CORS

Path

/

Resource ID

efg456

Methods (1)

Delete

Create method

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

Panel Resources menunjukkan struktur API yang dibuat sebagai pohon node. Metode API yang didefinisikan pada setiap sumber daya adalah tepi pohon. Ketika sumber daya dipilih, semua metodenya tercantum dalam tabel Metode di sebelah kanan. Ditampilkan dengan setiap metode adalah jenis metode, jenis integrasi, jenis otorisasi, dan persyaratan kunci API.

- Untuk melihat detail metode, untuk memodifikasi pengaturannya, atau untuk menguji pemanggilan metode, pilih nama metode dari daftar metode atau pohon sumber daya. Di sini, kami memilih POST `/pets` metode sebagai ilustrasi:

API Gateway > APIs > Resources - PetStore (acbd1234)

Resources

[API actions](#)
[Deploy API](#)

[Create resource](#)

- /
 - GET
 - /pets
 - GET
 - OPTIONS
 - POST**
 - /petid
 - GET
 - OPTIONS

/pets - POST - Method execution

[Update documentation](#)
[Delete](#)

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

Resource ID
aaa111

```

graph LR
    Client[Client] -- Method request --> AG[API Gateway]
    AG -- Integration request --> HTTP[HTTP integration]
    HTTP -- Integration response --> AG
    AG -- Method response --> Client
            
```

[<](#)
[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

[Edit](#)

Panel eksekusi Metode yang dihasilkan menyajikan tampilan logis dari struktur dan perilaku metode yang dipilih (POST /pets).

Permintaan Metode dan respons Metode mewakili antarmuka API dengan frontend, dan permintaan Integrasi serta respons Integrasi mewakili antarmuka API dengan backend.

Klien menggunakan API untuk mengakses fitur backend melalui permintaan Metode. API Gateway menerjemahkan permintaan klien, jika perlu, ke dalam formulir yang dapat diterima oleh backend dalam permintaan Integrasi sebelum meneruskan permintaan masuk ke backend. Permintaan yang ditransformasikan dikenal sebagai permintaan integrasi. Demikian pula, backend mengembalikan respons ke API Gateway dalam respons Integrasi. API Gateway kemudian merutekan ke Method Response sebelum mengirimnya ke klien. Sekali lagi, jika perlu, API Gateway dapat memetakan data respons backend ke formulir yang diharapkan oleh klien.

Untuk POST metode pada sumber daya API, payload permintaan metode dapat diteruskan ke permintaan integrasi tanpa modifikasi, jika payload permintaan metode memiliki format yang sama dengan payload permintaan integrasi.

Permintaan GET / metode menggunakan tipe MOCK integrasi dan tidak terikat pada titik akhir backend nyata. Respons Integrasi yang sesuai diatur untuk mengembalikan halaman HTML statis. Ketika metode dipanggil, API Gateway hanya menerima permintaan dan segera mengembalikan respons integrasi yang dikonfigurasi ke klien melalui respons Metode. Anda dapat menggunakan integrasi tiruan untuk menguji API tanpa memerlukan titik akhir backend. Anda juga dapat menggunakannya untuk menyajikan respons lokal, yang dihasilkan dari template pemetaan badan respons.

Sebagai pengembang API, Anda mengontrol perilaku interaksi frontend API Anda dengan mengonfigurasi permintaan metode dan respons metode. Anda mengontrol perilaku interaksi backend API Anda dengan menyiapkan permintaan integrasi dan respons integrasi. Ini melibatkan pemetaan data antara metode dan integrasi yang sesuai. Untuk saat ini, kami fokus pada pengujian API untuk memberikan pengalaman end-to-end pengguna.

6. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
7. Misalnya, untuk menguji POST /pets metode, masukkan **{"type": "dog", "price": 249.99}** payload berikut ke dalam badan Permintaan, lalu pilih Uji.



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 }
```

Input menentukan atribut hewan peliharaan yang ingin kita tambahkan ke daftar hewan peliharaan di situs PetStore web.

8. Hasil ditampilkan sebagai berikut:

/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-

Entri Log output menunjukkan perubahan status dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode. Ini dapat berguna untuk memecahkan masalah kesalahan pemetaan yang menyebabkan permintaan gagal. Dalam contoh ini, tidak ada pemetaan yang diterapkan: payload permintaan metode diteruskan melalui permintaan integrasi ke backend dan, demikian pula, respons backend diteruskan melalui respons integrasi ke respons metode.

Untuk menguji API menggunakan klien selain `test-invoke-request` fitur API Gateway, Anda harus terlebih dahulu menerapkan API ke tahap.

9. Untuk menerapkan API sampel, pilih `Deploy API`.

[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. Untuk Stage, pilih New stage, dan kemudian enter **test**.
11. (Opsional) Untuk Deskripsi, masukkan deskripsi.
12. Pilih Deploy.
13. Di panel Tahapan yang dihasilkan, di bawah Detail tahap, URL Invoke menampilkan URL untuk memanggil permintaan metode API. GET /

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)

- Pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Respons yang berhasil mengembalikan hasil, yang dihasilkan dari template pemetaan dalam respons integrasi.
- Di panel navigasi Tahapan, perluas tahap pengujian, pilih GET on/pets/{petId}, lalu salin nilai URL Invoke dari. `https://api-id.execute-api.region.amazonaws.com/test/pets/{petId}` {petId} singkatan dari variabel jalur.

Rekatkan nilai URL Invoke (diperoleh pada langkah sebelumnya) ke bilah alamat browser, ganti {petId} dengan, misalnya1, dan tekan Enter untuk mengirimkan permintaan. Respons 200 OK harus kembali dengan muatan JSON berikut:

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

Memanggil metode API seperti yang ditunjukkan dimungkinkan karena jenis Otorisasi disetel ke. NONE Jika AWS_IAM otorisasi digunakan, Anda akan menandatangani permintaan menggunakan protokol [Signature Version 4 \(SigV4\)](#) atau [Signature Version 4a \(Sigv4a\)](#). Untuk contoh permintaan seperti itu, lihat [the section called “Tutorial: Membuat REST API dengan integrasi non-proxy HTTP”](#).

Pilih tutorial integrasi HTTP

Untuk membangun API dengan integrasi HTTP, Anda dapat menggunakan integrasi proxy HTTP atau integrasi kustom HTTP.

Dalam integrasi proxy HTTP, Anda hanya perlu mengatur metode HTTP dan URI endpoint HTTP, sesuai dengan persyaratan backend. Kami menyarankan Anda menggunakan integrasi proxy HTTP, bila memungkinkan, untuk memanfaatkan pengaturan API yang disederhanakan.

Anda mungkin ingin menggunakan integrasi kustom HTTP jika Anda perlu mengubah data permintaan klien untuk backend atau mengubah data respons backend untuk klien.

Topik

- [Tutorial: Membuat REST API dengan integrasi proxy HTTP](#)
- [Tutorial: Membuat REST API dengan integrasi non-proxy HTTP](#)

Tutorial: Membuat REST API dengan integrasi proxy HTTP

Integrasi proxy HTTP adalah mekanisme sederhana, kuat, dan serbaguna untuk membangun API yang memungkinkan aplikasi web mengakses beberapa sumber daya atau fitur dari titik akhir HTTP terintegrasi, misalnya seluruh situs web, dengan pengaturan yang efisien dari satu metode API.

Dalam integrasi proxy HTTP, API Gateway meneruskan permintaan metode yang dikirimkan klien ke backend. Data permintaan yang diteruskan mencakup header permintaan, parameter string kueri, variabel jalur URL, dan payload. Titik akhir HTTP backend atau server web mem-parsing data permintaan yang masuk untuk menentukan respons yang dikembalikan. Integrasi proxy HTTP membuat klien dan backend berinteraksi secara langsung tanpa intervensi dari API Gateway setelah metode API disiapkan, kecuali untuk masalah yang diketahui seperti karakter yang tidak didukung, yang tercantum di dalamnya. [the section called “Catatan penting”](#)

Dengan sumber daya proxy yang mencakup semua{proxy+}, dan ANY kata kerja catch-all untuk metode HTTP, Anda dapat menggunakan integrasi proxy HTTP untuk membuat API dari satu metode API. Metode ini mengekspos seluruh rangkaian sumber daya HTTP yang dapat diakses publik dan operasi situs web. Ketika server web backend membuka lebih banyak sumber daya untuk akses publik, klien dapat menggunakan sumber daya baru ini dengan penyiapan API yang sama. Untuk mengaktifkan ini, pengembang situs web harus berkomunikasi dengan jelas kepada pengembang klien apa sumber daya baru dan operasi apa yang berlaku untuk masing-masing sumber daya tersebut.

Sebagai pengantar singkat, tutorial berikut menunjukkan integrasi proxy HTTP. Dalam tutorial, kita membuat API menggunakan konsol API Gateway untuk berintegrasi dengan PetStore situs web melalui sumber daya proxy generik{proxy+}, dan membuat placeholder metode HTTP dari. ANY

Topik

- [Membuat API dengan integrasi proxy HTTP menggunakan konsol API Gateway](#)
- [Uji API dengan integrasi proxy HTTP](#)

Membuat API dengan integrasi proxy HTTP menggunakan konsol API Gateway

Prosedur berikut memandu Anda melalui langkah-langkah untuk membuat dan menguji API dengan sumber daya proxy untuk backend HTTP menggunakan konsol API Gateway. Backend HTTP adalah PetStore situs web (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>) dari [Tutorial: Membuat REST API dengan integrasi non-proxy HTTP](#), di mana tangkapan layar digunakan sebagai alat bantu visual untuk mengilustrasikan elemen UI API Gateway. Jika Anda baru menggunakan konsol API Gateway untuk membuat API, Anda mungkin ingin mengikuti bagian itu terlebih dahulu.

Untuk membuat API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **HTTPProxyAPI**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Pada langkah ini, Anda membuat jalur sumber daya proxy dari{proxy+}. Ini adalah placeholder dari salah satu titik akhir backend di bawah. <http://petstore-demo-endpoint.execute-api.com/> Misalnya, bisa jadi [petstore/pets](http://petstore-demo-endpoint.execute-api.com/petstore/pets), dan [petstore/pets/{petId}](http://petstore-demo-endpoint.execute-api.com/petstore/pets/{petId}). API

Gateway membuat ANY metode saat Anda membuat {proxy+} sumber daya dan berfungsi sebagai placeholder untuk salah satu kata kerja HTTP yang didukung pada waktu berjalan.

Untuk membuat sumber daya/{proxy+}

1. Pilih API Anda.
2. Di panel navigasi utama, pilih Resources.
3. Pilih Buat sumber daya.
4. Aktifkan sumber daya Proxy.
5. Pertahankan jalur Sumber Daya sebagai/.
6. Untuk Nama sumber daya, masukkan **{proxy+}**.
7. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
8. Pilih Buat sumber daya.

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

Resource name

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

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

Pada langkah ini, Anda mengintegrasikan ANY metode dengan titik akhir HTTP backend, menggunakan integrasi proxy. Dalam integrasi proxy, API Gateway meneruskan permintaan metode yang dikirimkan klien ke backend tanpa intervensi dari API Gateway.

Untuk membuat **ANY** metode

1. Pilih sumber daya/{proxy+}.
2. Pilih metode APAPUN.
3. Di bawah simbol peringatan, pilih Edit integrasi. Anda tidak dapat menerapkan API yang memiliki metode tanpa integrasi.
4. Untuk jenis Integrasi, pilih HTTP.
5. Aktifkan integrasi proxy HTTP.
6. Untuk metode HTTP, pilih APAPUN.
7. Untuk URL Endpoint, masukkan **http://petstore-demo-endpoint.execute-api.com/{proxy}**.
8. Pilih Simpan.

Uji API dengan integrasi proxy HTTP

Apakah permintaan klien tertentu berhasil tergantung pada hal berikut:

- Jika backend telah membuat titik akhir backend yang sesuai tersedia dan, jika demikian, telah memberikan izin akses yang diperlukan.
- Jika klien memberikan masukan yang benar.

Misalnya, PetStore API yang digunakan di sini tidak mengekspos /petstore sumber daya. Dengan demikian, Anda mendapatkan 404 Resource Not Found respons yang berisi pesan kesalahanCannot GET /petstore.

Selain itu, klien harus dapat menangani format output backend untuk mengurai hasil dengan benar. API Gateway tidak memediasi untuk memfasilitasi interaksi antara klien dan backend.

Untuk menguji API yang terintegrasi dengan PetStore situs web menggunakan integrasi proxy HTTP melalui sumber daya proxy

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk jenis Metode, pilihGET.
3. Untuk Path, di bawah proxy, masukkan**petstore/pets**.
4. Untuk string Query, masukkan**type=fish**.
5. Pilih Uji.

[<](#) [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

Karena situs web backend mendukung GET /petstore/pets?type=fish permintaan, ia mengembalikan respons yang berhasil serupa dengan yang berikut ini:

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

Jika Anda mencoba menelepon `GET /petstore`, Anda mendapatkan `404` respons dengan pesan kesalahan `Cannot GET /petstore`. Ini karena backend tidak mendukung operasi yang ditentukan. Jika Anda menelepon `GET /petstore/pets/1`, Anda mendapatkan `200 OK` respons dengan muatan berikut, karena permintaan didukung oleh PetStore situs web.

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

Anda juga dapat menggunakan browser untuk menguji API Anda. Terapkan API Anda dan kaitkan ke panggung untuk membuat URL Invoke API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **test**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Sekarang klien dapat menghubungi API Anda.

Untuk menjalankan API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Di panel navigasi utama, pilih Stage.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Masukkan URL pemanggilan API Anda di browser web.

URL lengkap akan terlihat seperti `https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

Browser Anda mengirimkan GET permintaan ke API.

5. Hasilnya harus sama dengan yang dikembalikan saat Anda menggunakan Test di konsol API Gateway.

Tutorial: Membuat REST API dengan integrasi non-proxy HTTP

Dalam tutorial ini, Anda membuat API dari awal menggunakan konsol Amazon API Gateway. Anda dapat menganggap konsol sebagai studio desain API dan menggunakannya untuk cakupan fitur API, bereksperimen dengan perilakunya, membangun API, dan menerapkan API Anda secara bertahap.

Topik

- [Buat API dengan integrasi kustom HTTP](#)
- [\(Opsional\) Parameter permintaan peta](#)

Buat API dengan integrasi kustom HTTP

Bagian ini memandu Anda melalui langkah-langkah untuk membuat sumber daya, mengekspos metode pada sumber daya, mengonfigurasi metode untuk mencapai perilaku API yang diinginkan, dan untuk menguji dan menerapkan API.

Pada langkah ini, Anda membuat API kosong. Dalam langkah-langkah berikut, Anda membuat sumber daya dan metode untuk menghubungkan API Anda ke `http://petstore-demo-endpoint.execute-api.com/petstore/pets` titik akhir, menggunakan integrasi HTTP non-proxy.

Untuk membuat API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **HTTPNonProxyAPI**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.

5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Pohon Resources menunjukkan sumber daya root (/) tanpa metode apa pun. Dalam latihan ini, kita akan membangun API dengan integrasi kustom HTTP dari PetStore situs web (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>.) Untuk tujuan ilustrasi, kami akan membuat /pets sumber daya sebagai anak dari root dan mengekspos metode GET pada sumber daya ini bagi klien untuk mengambil daftar item Hewan Peliharaan yang tersedia dari situs web. PetStore

Untuk membuat sumber daya /pets

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai /.
4. Untuk Nama sumber daya, masukkan **pets**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Pada langkah ini, Anda membuat GET metode pada sumber daya /pets. GETMetode ini terintegrasi dengan situs <http://petstore-demo-endpoint.execute-api.com/petstore/pets> web. Opsi lain untuk metode API termasuk yang berikut:

- POST, terutama digunakan untuk membuat sumber daya anak.
- PUT, terutama digunakan untuk memperbarui sumber daya yang ada (dan, meskipun tidak disarankan, dapat digunakan untuk membuat sumber daya anak).
- DELETE, digunakan untuk menghapus sumber daya.
- PATCH, digunakan untuk memperbarui sumber daya.
- HEAD, terutama digunakan dalam skenario pengujian. Ini sama dengan GET tetapi tidak mengembalikan representasi sumber daya.
- OPTIONS, yang dapat digunakan oleh penelepon untuk mendapatkan informasi tentang opsi komunikasi yang tersedia untuk layanan target.

Untuk metode HTTP permintaan integrasi, Anda harus memilih salah satu yang didukung oleh backend. Untuk HTTP atau `Mock integration`, masuk akal bahwa permintaan metode dan permintaan integrasi menggunakan kata kerja HTTP yang sama. Untuk jenis integrasi lainnya, permintaan metode kemungkinan akan menggunakan kata kerja HTTP yang berbeda dari permintaan integrasi. Misalnya, untuk memanggil fungsi Lambda, permintaan integrasi harus digunakan POST untuk memanggil fungsi, sedangkan permintaan metode dapat menggunakan kata kerja HTTP apa pun tergantung pada logika fungsi Lambda.

Untuk membuat **GET** metode pada sumber daya `/pets`

1. Pilih sumber daya `/pets`.
2. Pilih metode Buat.
3. Untuk tipe Metode, pilih GET.
4. Untuk jenis Integrasi, pilih Integrasi HTTP.
5. Matikan integrasi proxy HTTP.
6. Untuk metode HTTP, pilih GET.
7. Untuk URL Endpoint, masukkan **`http://petstore-demo-endpoint.execute-api.com/petstore/pets`**.

PetStore Situs web ini memungkinkan Anda untuk mengambil daftar Pet item berdasarkan jenis hewan peliharaan, seperti “Dog” atau “Cat”, pada halaman tertentu.

8. Untuk penanganan Konten, pilih Passthrough.
9. Pilih parameter string kueri URL.

Situs PetStore web menggunakan parameter string `type` dan `page` kueri untuk menerima input. Anda menambahkan parameter string kueri ke permintaan metode dan memetakannya ke dalam parameter string kueri yang sesuai dari permintaan integrasi.

10. Untuk menambahkan parameter string kueri, lakukan hal berikut:
 - a. Pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan **`type`**
 - c. Tetap Diperlukan dan Caching dimatikan.

Ulangi langkah sebelumnya untuk membuat string kueri tambahan dengan nama **`page`**.

11. Pilih metode Buat.

Klien sekarang dapat menyediakan jenis hewan peliharaan dan nomor halaman sebagai parameter string kueri saat mengirimkan permintaan. Parameter input ini harus dipetakan ke dalam parameter string kueri integrasi untuk meneruskan nilai input ke PetStore situs web kami di backend.

Untuk memetakan parameter input ke permintaan Integrasi

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter string kueri URL, lalu lakukan hal berikut:
 - a. Pilih Tambahkan parameter string kueri.
 - b. Untuk Nama, masukkan **type**.
 - c. Untuk Dipetakan dari, masukkan **method.request.querystring.type**
 - d. Tetap caching dimatikan.
 - e. Pilih Tambahkan parameter string kueri.
 - f. Untuk Nama, masukkan **page**.
 - g. Untuk Dipetakan dari, masukkan **method.request.querystring.page**
 - h. Tetap caching dimatikan.
3. Pilih Simpan.

Untuk menguji API

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Query, masukkan **type=Dog&page=2**.
3. Pilih Uji.

Hasilnya mirip dengan yang berikut:

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

Headers

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

Client certificate

Test

/pets - GET method test results

Request

/pets?type=Dog&page=2

Latency ms

10

Status

200

Response body

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

Sekarang setelah pengujian berhasil, kami dapat menerapkan API untuk membuatnya tersedia untuk umum.

4. Pilih Deploy API.
5. Untuk Stage, pilih New stage.
6. Untuk nama Panggung, masukkan **Prod**.
7. (Opsional) Untuk Deskripsi, masukkan deskripsi.
8. Pilih Deploy.
9. (Opsional) Di bawah detail Tahap, untuk Invoke URL, Anda dapat memilih ikon salin untuk menyalin URL pemanggilan API Anda. Anda dapat menggunakan ini dengan alat seperti [Postman](#) dan [cURL](#) untuk menguji API Anda.

Jika Anda menggunakan SDK untuk membuat klien, Anda dapat memanggil metode yang diekspos oleh SDK untuk menandatangani permintaan. Untuk detail implementasi, lihat [AWS SDK](#) yang Anda pilih.

 Note

Saat perubahan dilakukan pada API Anda, Anda harus menerapkan ulang API agar fitur baru atau yang diperbarui tersedia sebelum menjalankan URL permintaan lagi.

(Opsional) Parameter permintaan peta

Parameter permintaan peta untuk API Gateway API

Tutorial ini menunjukkan cara membuat parameter jalur `{petId}` pada permintaan metode API untuk menentukan ID item, memetakannya ke parameter `{id}` jalur di URL permintaan integrasi, dan mengirim permintaan ke titik akhir HTTP.

 Note

Jika Anda memasukkan huruf yang salah, seperti huruf kecil, bukan huruf besar, ini akan menyebabkan kesalahan nanti dalam penelusuran.

Langkah 1: Buat sumber daya

Pada langkah ini, Anda membuat sumber daya dengan parameter jalur `{PetId}`.

Untuk membuat sumber daya `{PetId}`

1. Pilih sumber daya `/pets`, lalu pilih **Create resource**.
2. Matikan sumber daya **Proxy**.
3. Untuk jalur Sumber Daya, pilih `/pets/`.
4. Untuk Nama sumber daya, masukkan **`{petId}`**.

Gunakan kurawal kurawal (`{ }`) di sekitar `petId` sehingga `/pets/ {PetId}` ditampilkan.

5. Tetap nonaktifkan **CORS (Cross Origin Resource Sharing)**.
6. Pilih **Buat sumber daya**.

Langkah 2: Buat dan uji metode

Pada langkah ini, Anda membuat GET metode dengan parameter {petId} jalur.

Untuk mengatur metode GET

1. Pilih sumber daya/{PeTiD}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Integrasi HTTP.
4. Matikan integrasi proxy HTTP.
5. Untuk metode HTTP, pilih GET.
6. Untuk URL Endpoint, masukkan **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}**
7. Untuk penanganan Konten, pilih Passthrough.
8. Tetap aktifkan batas waktu default.
9. Pilih metode Buat.

Sekarang Anda memetakan parameter {petId} path yang baru saja Anda buat ke parameter {id} path di URL endpoint HTTP dari permintaan integrasi. URL titik akhir HTTP adalah **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}**.

Untuk memetakan parameter **{petId}** jalur

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL.
3. API Gateway membuat parameter jalur untuk permintaan integrasi bernama PeTiD, namun parameter jalur ini tidak valid untuk URL titik akhir HTTP yang Anda tetapkan sebagai integrasi backend. Titik akhir HTTP digunakan {id} sebagai parameter jalur. Untuk Nama, hapus PeTiD dan masukkan. **id**

Ini memetakan parameter jalur permintaan metode petId ke parameter jalur permintaan integrasi **id**.

4. Pilih Simpan.

Sekarang Anda menguji metodenya.

Untuk menguji metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path for PeTiD, masukkan. **4**
3. Pilih Uji.

Jika berhasil, badan Response menampilkan yang berikut ini:

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

Langkah 3: Menyebarkan API

Pada langkah ini, Anda menerapkan API sehingga Anda dapat mulai memanggilnya di luar konsol API Gateway.

Untuk menerapkan API

1. Pilih Deploy API.
2. Untuk Stage, pilih Prod.
3. (Opsional) Untuk Deskripsi, masukkan deskripsi.
4. Pilih Deploy.

Langkah 4: Uji API

Pada langkah ini, Anda pergi ke luar konsol API Gateway dan menggunakan API Anda untuk mengakses titik akhir HTTP.

1. Di panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Seharusnya terlihat seperti ini:

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

3. Masukkan URL ini di kotak alamat tab browser baru dan tambahkan `/pets/4` ke URL sebelum Anda mengirimkan permintaan Anda.
4. Browser akan mengembalikan yang berikut:

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

Langkah selanjutnya

Anda dapat menyesuaikan API lebih lanjut dengan mengaktifkan validasi permintaan, mengubah data, atau membuat respons gateway khusus.

Untuk menjelajahi lebih banyak cara untuk menyesuaikan API Anda, lihat tutorial berikut:

- Untuk informasi selengkapnya tentang validasi permintaan, lihat [Siapkan validasi permintaan dasar di API Gateway](#).
- Untuk informasi tentang cara mengubah payload permintaan dan respons, lihat [Tutorial: Memodifikasi permintaan integrasi dan respon untuk integrasi AWS layanan](#).
- Untuk informasi tentang cara membuat respons gateway kustom, lihat [Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway](#).

Tutorial: Buat REST API dengan integrasi pribadi

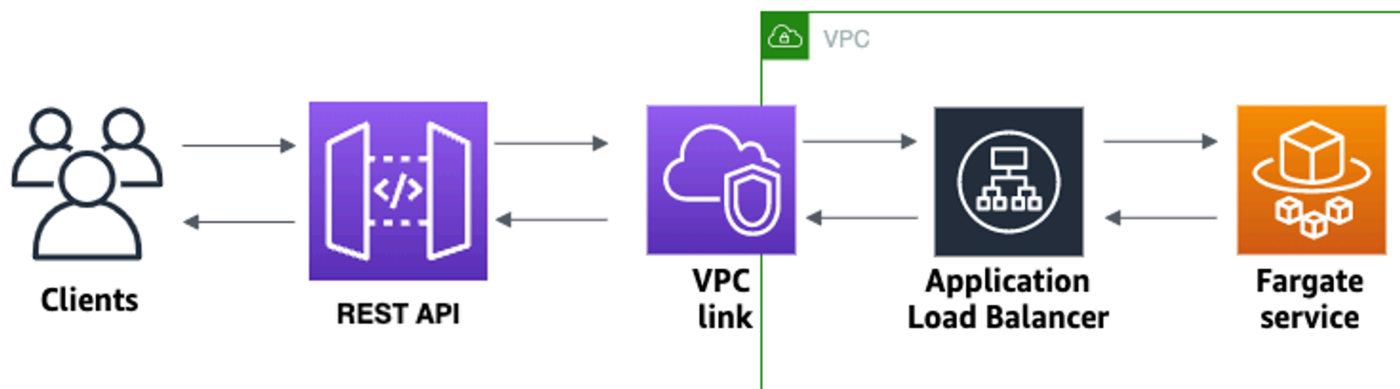
Dalam tutorial ini, Anda membuat REST API yang terhubung ke layanan Amazon ECS yang berjalan di Amazon VPC. Klien di luar VPC Amazon Anda dapat menggunakan API untuk mengakses layanan Amazon ECS Anda.

Tutorial ini membutuhkan waktu sekitar satu jam untuk menyelesaikannya. Pertama, Anda menggunakan CloudFormation template untuk membuat Amazon VPC dan layanan Amazon ECS. Kemudian Anda menggunakan konsol API Gateway untuk membuat tautan VPC V2. Tautan VPC memungkinkan API Gateway untuk mengakses layanan Amazon ECS yang berjalan di VPC Amazon Anda. Selanjutnya, Anda membuat REST API yang menggunakan tautan VPC V2 untuk terhubung ke layanan Amazon ECS Anda. Terakhir, Anda menguji API Anda.

Saat Anda menjalankan REST API, API Gateway merutekan permintaan ke layanan Amazon ECS Anda melalui tautan VPC V2, lalu mengembalikan respons dari layanan.

Note

Tutorial ini sebelumnya didukung untuk HTTP APIs, dan sekarang didukung untuk REST APIs menggunakan VPC link V2.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Topik

- [Langkah 1: Buat layanan Amazon ECS](#)
- [Langkah 2: Buat tautan VPC](#)
- [Langkah 3: Buat REST API](#)
- [Langkah 4: Uji API Anda](#)
- [Langkah 5: Menerapkan API Anda](#)
- [Langkah 6: Hubungi API Anda](#)
- [Langkah 7: Bersihkan](#)

Langkah 1: Buat layanan Amazon ECS

Amazon ECS adalah layanan manajemen kontainer yang memudahkan untuk menjalankan, menghentikan, dan mengelola kontainer Docker di cluster. Dalam tutorial ini, Anda menjalankan cluster Anda pada infrastruktur tanpa server yang dikelola oleh Amazon ECS.

Unduh dan unzip [CloudFormation template ini](#), yang membuat semua dependensi untuk layanan, termasuk VPC Amazon. Anda menggunakan template untuk membuat layanan Amazon ECS yang menggunakan Application Load Balancer.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **rest-api-private-integrations-tutorial** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation menyediakan layanan ECS, yang dapat memakan waktu beberapa menit. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat tautan VPC

Tautan VPC memungkinkan API Gateway untuk mengakses sumber daya pribadi di VPC Amazon. Anda menggunakan tautan VPC untuk memungkinkan klien mengakses layanan Amazon ECS Anda melalui REST API Anda.

Untuk membuat tautan VPC

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada panel navigasi utama, pilih tautan VPC dan kemudian pilih Buat.

Anda mungkin perlu memilih ikon menu untuk membuka panel navigasi utama.

3. Untuk Pilih versi tautan VPC, pilih tautan VPC V2.
4. Untuk Nama, masukkan **private-integrations-tutorial**.

5. Untuk VPC, pilih VPC yang Anda buat di langkah 1. Nama harus dimulai dengan `RestApiStack`.
6. Untuk Subnet, pilih dua subnet pribadi di VPC Anda. Nama mereka diakhiri dengan `PrivateSubnet`
7. Untuk grup Keamanan, pilih ID Grup yang dimulai dengan `private-integrations-tutorial` dan memiliki deskripsi `RestApiStack/RestApiTutorialService/Service/SecurityGroup`.
8. Pilih Buat.

Setelah Anda membuat tautan VPC V2, API Gateway menyediakan Antarmuka Jaringan Elastis untuk mengakses VPC Anda. Prosesnya bisa memakan waktu beberapa menit. Sementara itu, Anda dapat membuat API Anda.

Langkah 3: Buat REST API

REST API menyediakan titik akhir HTTP untuk layanan Amazon ECS Anda.

Untuk membuat REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Create API, lalu untuk REST API, pilih Build.
3. Untuk Nama, masukkan **private-integration-api**.
4. Untuk jenis alamat IP, pilih IPv4.
5. Pilih Buat API.

Setelah membuat API, Anda membuat metode.

6. Pilih metode Create, dan kemudian lakukan hal berikut:
 - a. Untuk jenis Metode, pilih `GET`.
 - b. Untuk jenis Integrasi, pilih tautan VPC.
 - c. Aktifkan integrasi proxy VPC.
 - d. Untuk metode HTTP, pilih `GET`.
 - e. Untuk tautan VPC, pilih tautan VPC V2 yang Anda buat di langkah sebelumnya.
 - f. Untuk target Integrasi, masukkan penyeimbang beban yang Anda buat dengan CloudFormation template di Langkah 1. Namanya harus dimulai dengan istirahat-

- g. Untuk URL Endpoint, masukkan `http://private-integrations-tutorial.com`.

URL digunakan untuk mengatur Host header permintaan integrasi. Dalam hal ini, header host adalah **private-integrations-tutorial**.

- h. Pilih metode Buat.

Dengan integrasi proxy, API siap untuk diuji.

Langkah 4: Uji API Anda

Selanjutnya, Anda menguji pemanggilan metode API.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
4. Pilih Tes

Verifikasi bahwa respons API Anda adalah pesan selamat datang yang memberi tahu Anda bahwa aplikasi Anda berjalan di Amazon ECS.

Langkah 5: Menerapkan API Anda

Selanjutnya, Anda menerapkan API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **Prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Langkah 6: Hubungi API Anda

Setelah API Anda di-deploy, Anda dapat memanggilnya.

Untuk memanggil API Anda

1. Masukkan URL pemanggilan di browser web.

URL lengkap akan terlihat seperti `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Browser Anda mengirimkan GET permintaan ke API.

2. Verifikasi bahwa respons API Anda adalah pesan selamat datang yang memberi tahu Anda bahwa aplikasi Anda berjalan di Amazon ECS.

Jika Anda melihat pesan selamat datang, Anda berhasil membuat layanan Amazon ECS yang berjalan di VPC Amazon, dan Anda menggunakan API REST API Gateway API dengan tautan VPC V2 untuk mengakses layanan Amazon ECS.

Langkah 7: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus tautan VPC V2, CloudFormation stack, dan REST API Anda.

Untuk menghapus REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada API salaman, pilih API. Pilih Tindakan, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus tautan VPC

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih tautan VPC.
3. Pilih tautan VPC Anda, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Tutorial: Buat REST API dengan AWS integrasi

Baik [Tutorial: Buat REST API dengan integrasi non-proxy Lambda](#) topik [Tutorial: Buat REST API dengan integrasi proxy Lambda](#) maupun topik menjelaskan cara membuat API Gateway API untuk mengekspos fungsi Lambda terintegrasi. Selain itu, Anda dapat membuat API Gateway API untuk mengekspos AWS layanan lain, seperti Amazon SNS, Amazon S3, Amazon Kinesis, dan bahkan. AWS Lambda Hal ini dimungkinkan oleh AWS integrasi. Integrasi Lambda atau integrasi proxy Lambda adalah kasus khusus, di mana pemanggilan fungsi Lambda diekspos melalui API Gateway API.

Semua dukungan AWS layanan yang didedikasikan APIs untuk mengekspos fitur mereka. Namun, protokol aplikasi atau antarmuka pemrograman cenderung berbeda dari layanan ke layanan. API Gateway API dengan AWS integrasi memiliki keuntungan menyediakan protokol aplikasi yang konsisten bagi klien Anda untuk mengakses berbagai AWS layanan.

Dalam panduan ini, kami membuat API untuk mengekspos Amazon SNS. Untuk contoh lebih lanjut tentang mengintegrasikan API dengan AWS layanan lain, lihat [Tutorial dan lokakarya Amazon API Gateway](#).

Berbeda dengan integrasi proxy Lambda, tidak ada integrasi proxy yang sesuai untuk layanan lain AWS . Oleh karena itu, metode API terintegrasi dengan satu AWS tindakan. Untuk fleksibilitas lebih, mirip dengan integrasi proxy, Anda dapat mengatur integrasi proxy Lambda. Fungsi Lambda kemudian mem-parsing dan memproses permintaan untuk tindakan lain. AWS

API Gateway tidak mencoba lagi saat titik akhir habis. Pemanggil API harus menerapkan logika coba lagi untuk menangani batas waktu titik akhir.

Panduan ini dibangun di atas instruksi dan konsep di. [Tutorial: Buat REST API dengan integrasi non-proxy Lambda](#) Jika Anda belum menyelesaikan panduan itu, kami sarankan Anda melakukannya terlebih dahulu.

Topik

- [Prasyarat](#)
- [Langkah 1: Buat peran eksekusi proxy AWS layanan](#)
- [Langkah 2: Buat sumber daya](#)
- [Langkah 3: Buat metode GET](#)
- [Langkah 4: Tentukan pengaturan metode dan uji metodenya](#)
- [Langkah 5: Menerapkan API](#)

- [Langkah 6: Uji API](#)
- [Langkah 7: Bersihkan](#)

Prasyarat

Sebelum memulai panduan ini, lakukan hal berikut:

1. Selesaikan langkah-langkah dalam [Siapkan untuk menggunakan API Gateway](#).
2. Buat API baru bernama `MyDemoAPI`. Untuk informasi selengkapnya, lihat [Tutorial: Membuat REST API dengan integrasi non-proxy HTTP](#).
3. Terapkan API setidaknya sekali ke tahap bernama `test`. Untuk informasi selengkapnya, lihat [Menerapkan API](#) di [Pilih tutorial AWS Lambda integrasi](#).
4. Selesaikan langkah-langkah lainnya [Pilih tutorial AWS Lambda integrasi](#).
5. Buat setidaknya satu topik di Amazon Simple Notification Service (Amazon SNS). Anda akan menggunakan API yang diterapkan untuk mendapatkan daftar topik di Amazon SNS yang terkait dengan AWS akun Anda. Untuk mempelajari cara membuat topik di Amazon SNS, lihat [Membuat](#) Topik. (Anda tidak perlu menyalin topik ARN yang disebutkan di langkah 5.)

Langkah 1: Buat peran eksekusi proxy AWS layanan

Untuk mengizinkan API menjalankan tindakan Amazon SNS, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM. Pada langkah ini, Anda membuat peran IAM baru.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran **APIGatewaySNSProxyPolicy**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.

8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **AmazonSNSReadOnlyAccess** dan pilih Tambahkan izin.

 Note

Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Langkah 2: Buat sumber daya

Pada langkah ini, Anda membuat sumber daya yang memungkinkan proxy AWS layanan berinteraksi dengan AWS layanan.

Untuk membuat sumber daya

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih sumber daya root, /, yang diwakili oleh satu garis miring (/), lalu pilih Buat sumber daya.
4. Matikan sumber daya Proxy.
5. Pertahankan jalur Sumber Daya sebagai /.
6. Untuk Nama sumber daya, masukkan **mydemoawsproxy**.
7. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
8. Pilih Buat sumber daya.

Langkah 3: Buat metode GET

Pada langkah ini, Anda membuat metode GET yang memungkinkan proxy AWS layanan berinteraksi dengan AWS layanan.

Untuk membuat **GET** metode

1. Pilih sumber daya /mydemoawsproxy, lalu pilih Create method.

2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat topik Amazon SNS Anda.
5. Untuk Layanan AWS, pilih Amazon SNS.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk jenis tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **ListTopics**.
10. Untuk peran Eksekusi, masukkan peran ARN untuk **APIGatewaySNSProxyPolicy**
11. Pilih metode Buat.

Langkah 4: Tentukan pengaturan metode dan uji metodenya

Anda sekarang dapat menguji GET metode Anda untuk memverifikasi bahwa metode tersebut telah diatur dengan benar untuk mencantumkan topik Amazon SNS Anda.

Untuk menguji **GET** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Pilih Uji.

Hasilnya menampilkan respon yang mirip dengan berikut ini:

```
{
  "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"  
  }  
}  
}
```

Langkah 5: Menerapkan API

Pada langkah ini, Anda menerapkan API sehingga Anda dapat memanggilnya dari luar konsol API Gateway.

Untuk menerapkan API

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **test**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Langkah 6: Uji API

Pada langkah ini, Anda pergi ke luar konsol API Gateway dan menggunakan proxy AWS layanan Anda untuk berinteraksi dengan layanan Amazon SNS.

1. Di panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Seharusnya terlihat seperti ini:

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

3. Masukkan URL ke dalam kotak alamat tab browser baru.
4. Tambahkan /mydemoawsproxy sehingga URL terlihat seperti ini:

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

Jelajahi URL. Informasi yang mirip dengan berikut ini harus ditampilkan:

```
{"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"}}
```

Langkah 7: Bersihkan

Anda dapat menghapus sumber daya IAM yang diperlukan proxy AWS layanan untuk bekerja.

Warning

Jika Anda menghapus sumber daya IAM yang diandalkan oleh proxy AWS layanan, proxy AWS layanan itu dan apa pun APIs yang mengandalkannya tidak akan berfungsi lagi. Menghapus sumber daya IAM tidak dapat dibatalkan. Jika Anda ingin menggunakan sumber daya IAM lagi, Anda harus membuatnya kembali.

Untuk menghapus sumber daya IAM terkait

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di area Detail, pilih Peran.
3. Pilih APIGatewayAWSProxyExecRole, lalu pilih Tindakan Peran, Hapus Peran. Saat diminta, pilih Ya, Hapus.
4. Di area Detail, pilih Kebijakan.
5. Pilih APIGatewayAWSProxyExecPolicy, lalu pilih Tindakan Kebijakan, Hapus. Saat diminta, pilih Hapus.

Anda telah mencapai akhir dari panduan ini. Untuk diskusi lebih rinci tentang membuat API sebagai proxy AWS layanan, lihat [Tutorial: Buat REST API sebagai proxy Amazon S3](#), [Tutorial: Buat REST API kalkulator dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda](#), atau [Tutorial: Buat REST API sebagai proxy Amazon Kinesis](#).

Tutorial: Buat REST API kalkulator dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda

[the section called “Tutorial: Buat REST API dengan integrasi non-proxy Lambda”](#) Penggunaan Lambda Function integrasi secara eksklusif. Lambda Function integrasi adalah kasus khusus dari jenis AWS Service integrasi yang melakukan banyak pengaturan integrasi untuk Anda, seperti secara otomatis menambahkan izin berbasis sumber daya yang diperlukan untuk menjalankan fungsi Lambda. Di sini, dua dari tiga integrasi menggunakan AWS Service integrasi. Dalam jenis integrasi ini, Anda memiliki lebih banyak kontrol, tetapi Anda harus melakukan tugas secara manual seperti membuat dan menentukan peran IAM yang berisi izin yang sesuai.

Dalam tutorial ini, Anda akan membuat fungsi Calc Lambda yang mengimplementasikan operasi aritmatika dasar, menerima dan mengembalikan input dan output berformat JSON. Kemudian Anda akan membuat REST API dan mengintegrasikannya dengan fungsi Lambda dengan cara berikut:

1. Dengan mengekspos GET metode pada /calc sumber daya untuk menjalankan fungsi Lambda, memasok input sebagai parameter string kueri. (AWS Service integrasi)
2. Dengan mengekspos POST metode pada /calc sumber daya untuk menjalankan fungsi Lambda, memasok input dalam payload permintaan metode. (AWS Service integrasi)
3. Dengan mengekspos /calc/{operand1}/{operand2}/{operator} sumber daya GET bersarang untuk menjalankan fungsi Lambda, memasok input sebagai parameter jalur. (Lambda Function integrasi)

Selain mencoba tutorial ini, Anda mungkin ingin mempelajari [file definisi OpenAPI](#) untuk Calc API, yang dapat Anda impor ke API Gateway dengan mengikuti petunjuk di [the section called “OpenAPI”](#)

Topik

- [Buat peran IAM yang dapat diasumsikan](#)
- [Buat fungsi Calc Lambda](#)
- [Uji fungsi Calc Lambda](#)
- [Buat Calc API](#)
- [Integrasi 1: Buat GET metode dengan parameter kueri untuk memanggil fungsi Lambda](#)
- [Integrasi 2: Buat POST metode dengan payload JSON untuk memanggil fungsi Lambda](#)
- [Integrasi 3: Buat GET metode dengan parameter jalur untuk memanggil fungsi Lambda](#)

- [Definisi OpenAPI dari contoh API terintegrasi dengan fungsi Lambda](#)

Buat peran IAM yang dapat diasumsikan

Agar API dapat menjalankan fungsi Calc Lambda, Anda harus memiliki peran IAM yang dapat diasumsikan API Gateway, yang merupakan peran IAM dengan hubungan tepercaya berikut:

JSON

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

Peran yang Anda buat harus memiliki izin Lambda [InvokeFunction](#). Jika tidak, pemanggil API akan menerima 500 Internal Server Error respons. Untuk memberikan izin ini kepada peran ini, Anda akan melampirkan kebijakan IAM berikut:

JSON

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

Inilah cara untuk mencapai semua ini:

Buat peran IAM yang dapat diasumsikan API Gateway

1. Masuk ke konsol IAM.
2. Pilih Peran.
3. Pilih Buat Peran.
4. Di Pilih jenis entitas tepercaya, pilih Layanan AWS .
5. Di bawah Pilih layanan yang akan menggunakan di peran ini, pilih Lambda.
6. Pilih Berikutnya: Izin.
7. Pilih Buat Kebijakan.

Jendela konsol Create Policy baru akan terbuka. Di jendela itu, lakukan hal berikut:

- a. Di tab JSON, ganti kebijakan yang ada dengan yang berikut:

JSON

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

- b. Pilih Tinjau kebijakan.
- c. Di bawah Kebijakan Peninjauan, lakukan hal berikut:
 - i. Untuk Nama, ketik nama seperti **lambda_execute**.
 - ii. Pilih Buat Kebijakan.
8. Di jendela konsol Create Role asli, lakukan hal berikut:

- a. Di bawah Lampirkan kebijakan izin, pilih **lambda_execute** kebijakan Anda dari daftar tarik-turun.

Jika kebijakan tidak terlihat dalam daftar, pilih tombol refresh di bagian atas daftar. (Jangan me-refresh halaman browser!)

- b. Pilih Selanjutnya: Tag.
 - c. Pilih Selanjutnya: Tinjau.
 - d. Untuk nama Peran, ketikkan nama seperti **lambda_invoke_function_assume_apigw_role**.
 - e. Pilih Buat peran.
9. Pilih **lambda_invoke_function_assume_apigw_role** dari daftar peran Anda.
 10. Pilih tab Trust relationship.
 11. Pilih Edit trust relationship (Edit Hubungan Kepercayaan).
 12. Ganti kebijakan yang ada dengan yang berikut:

JSON

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

13. Pilih Perbarui Kebijakan Kepercayaan.

14. Catat peran ARN untuk peran yang baru saja Anda buat. Anda akan membutuhkannya nanti.

Buat fungsi **Calc** Lambda

Selanjutnya Anda akan membuat fungsi Lambda menggunakan konsol Lambda.

1. Di konsol Lambda, pilih Buat fungsi.
2. Pilih Penulis dari Scratch.
3. Untuk Nama, masukkan **Calc**.
4. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
5. Untuk semua opsi lain, gunakan pengaturan default.
6. Pilih Buat fungsi.
7. Salin fungsi Lambda berikut di runtime pilihan Anda dan tempelkan ke editor kode di konsol Lambda.

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. Di bawah Peran eksekusi, pilih Pilih peran yang ada.
9. Masukkan peran ARN untuk **lambda_invoke_function_assume_apigw_role** peran yang Anda buat sebelumnya.
10. Pilih Deploy.

Fungsi ini membutuhkan dua operan (adanb) dan operator (op) dari parameter event input. Input adalah objek JSON dari format berikut:

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

Fungsi ini mengembalikan hasil yang dihitung (c) dan masukan. Untuk input yang tidak valid, fungsi mengembalikan nilai null atau string "Invalid op" sebagai hasilnya. Outputnya adalah format JSON berikut:

```
{
  "a": "Number",
```

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

Anda harus menguji fungsi di konsol Lambda sebelum mengintegrasikannya dengan API di langkah berikutnya.

Uji fungsi **Calc** Lambda

Berikut cara menguji **Calc** fungsi Anda di konsol Lambda:

1. Pilih tab Uji.
2. Untuk nama acara pengujian, masukkan **calc2plus5**.
3. Ganti definisi acara pengujian dengan yang berikut:

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

4. Pilih Simpan.
5. Pilih Uji.
6. Perluas Hasil eksekusi: berhasil. Anda akan melihat yang berikut ini:

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

Buat **Calc** API

Prosedur berikut menunjukkan cara membuat API untuk fungsi **Calc** Lambda yang baru saja Anda buat. Di bagian berikutnya, Anda akan menambahkan sumber daya dan metode ke dalamnya.

Untuk membuat API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Untuk nama API, masukkan **LambdaCalc**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Integrasi 1: Buat **GET** metode dengan parameter kueri untuk memanggil fungsi Lambda

Dengan membuat GET metode yang meneruskan parameter string kueri ke fungsi Lambda, Anda mengaktifkan API untuk dipanggil dari browser. Pendekatan ini dapat berguna, terutama untuk APIs itu memungkinkan akses terbuka.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk langkah ini, Anda membuat sumber daya `/calc`.

Untuk membuat sumber daya `/calc`

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai `/`.
4. Untuk Nama sumber daya, masukkan **calc**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Dengan membuat GET metode yang meneruskan parameter string kueri ke fungsi Lambda, Anda mengaktifkan API untuk dipanggil dari browser. Pendekatan ini dapat berguna, terutama untuk APIs itu memungkinkan akses terbuka.

Dalam metode ini, Lambda mengharuskan POST permintaan digunakan untuk memanggil fungsi Lambda apa pun. Contoh ini menunjukkan bahwa metode HTTP dalam permintaan metode frontend dapat berbeda dari permintaan integrasi di backend.

Untuk membuat **GET** metode

1. Pilih sumber daya /calc, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
5. Untuk Layanan AWS, pilih Lambda.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Action, pilih Use path override. Opsi ini memungkinkan kita untuk menentukan ARN dari tindakan [Invoke](#) untuk menjalankan fungsi kita. Calc
9. Untuk Path override, masukkan **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations**. Untuk **account-id**, masukkan Akun AWS ID Anda. Untuk **us-east-2**, masukkan Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
10. Untuk peran Eksekusi, masukkan peran ARN untuk **lambda_invoke_function_assume_apigw_role**
11. Jangan mengubah pengaturan cache Credential dan batas waktu default.
12. Pilih Pengaturan permintaan Metode.
13. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header.

Pengaturan ini akan menyebabkan pesan kesalahan kembali jika klien tidak menentukan parameter yang diperlukan.

14. Pilih parameter string kueri URL.

Sekarang Anda mengatur parameter string kueri untuk metode GET pada sumber daya /calc sehingga dapat menerima input atas nama fungsi Lambda backend.

Untuk membuat parameter string kueri lakukan hal berikut:

- a. Pilih Tambahkan string kueri.
- b. Untuk Nama, masukkan **operand1**.
- c. Aktifkan Diperlukan.
- d. Tetap caching dimatikan.

Ulangi langkah yang sama dan buat string kueri bernama **operand2** dan string kueri bernama **operator**.

15. Pilih metode Buat.

Sekarang, Anda membuat template pemetaan untuk menerjemahkan string kueri yang disediakan klien ke payload permintaan integrasi seperti yang dipersyaratkan oleh fungsi. Calc Template ini memetakan tiga parameter string kueri yang dideklarasikan dalam permintaan Metode ke dalam nilai properti yang ditunjuk dari objek JSON sebagai masukan ke fungsi Lambda backend. Objek JSON yang diubah akan dimasukkan sebagai payload permintaan integrasi.

Untuk memetakan parameter input ke permintaan integrasi

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih template Pemetaan.
4. Pilih Tambahkan templat pemetaan.
5. Untuk jenis Konten, masukkan **application/json**.
6. Untuk badan Template, masukkan kode berikut:

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

7. Pilih Simpan.

Anda sekarang dapat menguji GET metode Anda untuk memverifikasi bahwa itu telah diatur dengan benar untuk menjalankan fungsi Lambda.

Untuk menguji **GET** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Query, masukkan **operand1=2&operand2=3&operator=+**.
3. Pilih Uji.

Hasilnya akan terlihat mirip dengan ini:

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

Integrasi 2: Buat **POST** metode dengan payload JSON untuk memanggil fungsi Lambda

Dengan membuat POST metode dengan payload JSON untuk memanggil fungsi Lambda, Anda membuatnya sehingga klien harus memberikan input yang diperlukan ke fungsi backend di badan permintaan. Untuk memastikan bahwa klien mengunggah data input yang benar, Anda akan mengaktifkan validasi permintaan pada payload.

Untuk membuat **POST** metode dengan payload JSON

1. Pilih sumber daya /calc, lalu pilih Create method.
2. Untuk jenis Metode, pilih POST.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
5. Untuk Layanan AWS, pilih Lambda.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.

8. Untuk tipe Action, pilih Use path override. Opsi ini memungkinkan kita untuk menentukan ARN dari tindakan [Invoke](#) untuk menjalankan fungsi kita. Calc
9. Untuk Path override, masukkan **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations**. Untuk **account-id**, masukkan Akun AWS ID Anda. Untuk **us-east-2**, masukkan Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
10. Untuk peran Eksekusi, masukkan peran ARN untuk **lambda_invoke_function_assume_apigw_role**
11. Jangan mengubah pengaturan cache Credential dan batas waktu default.
12. Pilih metode Buat.

Sekarang Anda membuat model input untuk menggambarkan struktur data input dan memvalidasi badan permintaan yang masuk.

Untuk membuat model input

1. Di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **input**.
4. Untuk jenis Konten, masukkan **application/json**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **\$default**.

5. Untuk skema Model, masukkan model berikut:

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

6. Pilih Buat model.

Anda sekarang membuat model output. Model ini menjelaskan struktur data dari output yang dihitung dari backend. Ini dapat digunakan untuk memetakan data respons integrasi ke model yang berbeda. Tutorial ini bergantung pada perilaku passthrough dan tidak menggunakan model ini.

Untuk membuat model keluaran

1. Pilih Buat model.
2. Untuk Nama, masukkan **output**.
3. Untuk jenis Konten, masukkan **application/json**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **\$default**.

4. Untuk skema Model, masukkan model berikut:

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

5. Pilih Buat model.

Anda sekarang membuat model hasil. Model ini menjelaskan struktur data dari data respons yang dikembalikan. Ini mereferensikan skema input dan output yang ditentukan dalam API Anda.

Untuk membuat model hasil

1. Pilih Buat model.
2. Untuk Nama, masukkan **result**.
3. Untuk jenis Konten, masukkan **application/json**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **\$default**.

4. Untuk skema Model, masukkan model berikut dengan skema Anda **restapi-id**. Anda **restapi-id** tercantum dalam tanda kurung di bagian atas konsol dalam alur berikut: 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. Pilih Buat model.

Anda sekarang mengonfigurasi permintaan metode metode POST Anda untuk mengaktifkan validasi permintaan pada badan permintaan yang masuk.

Untuk mengaktifkan validasi permintaan pada metode POST

1. Di panel navigasi utama, pilih Sumber daya, lalu pilih POST metode dari pohon sumber daya.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi isi.
4. Pilih Request body, lalu pilih Add model.
5. Untuk jenis Konten, masukkan **application/json**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **\$default**.

6. Untuk Model, pilih input.
7. Pilih Simpan.

Anda sekarang dapat menguji POST metode Anda untuk memverifikasi bahwa itu telah diatur dengan benar untuk menjalankan fungsi Lambda.

Untuk menguji **POST** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk badan Permintaan, masukkan payload JSON berikut.

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

3. Pilih Uji.

Anda akan melihat output berikut:

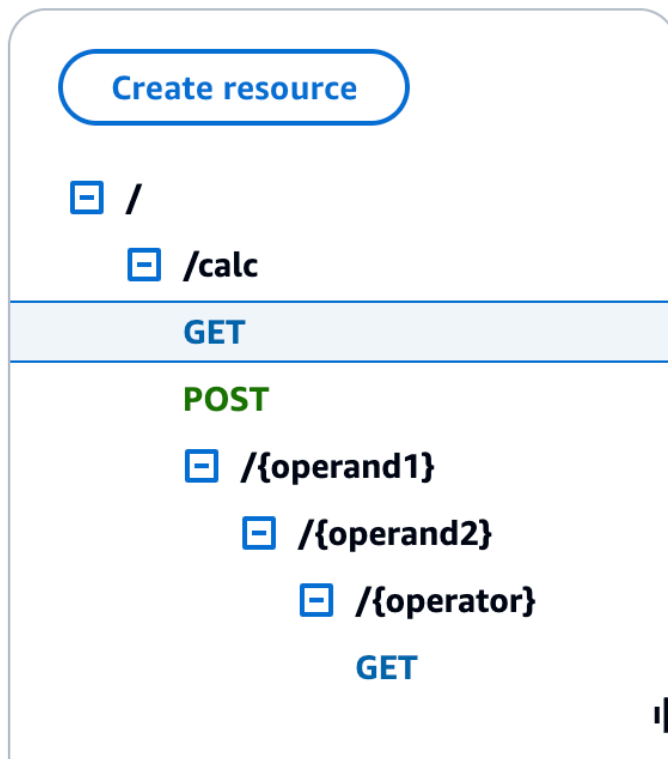
```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```

Integrasi 3: Buat **GET** metode dengan parameter jalur untuk memanggil fungsi Lambda

Sekarang Anda akan membuat GET metode pada sumber daya yang ditentukan oleh urutan parameter jalur untuk memanggil fungsi Lambda backend. Nilai parameter jalur menentukan data input ke fungsi Lambda. Anda akan menggunakan template pemetaan untuk memetakan nilai parameter jalur masuk ke payload permintaan integrasi yang diperlukan.

Struktur sumber daya API yang dihasilkan akan terlihat seperti ini:

Resources



Untuk membuat sumber daya `/operand1/operand2/operator`

1. Pilih Buat sumber daya.
2. Untuk jalur Sumber Daya, pilih `/calc`.
3. Untuk Nama sumber daya, masukkan **`{operand1}`**.
4. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
5. Pilih Buat sumber daya.
6. Untuk jalur Sumber Daya, pilih `/calc/{operand1}/`.
7. Untuk Nama sumber daya, masukkan **`{operand2}`**.
8. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
9. Pilih Buat sumber daya.
10. Untuk jalur Sumber Daya, pilih `/calc/{operand1}/{operand2}/`.
11. Untuk Nama sumber daya, masukkan **`{operator}`**.
12. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
13. Pilih Buat sumber daya.

Kali ini Anda akan menggunakan integrasi Lambda bawaan di konsol API Gateway untuk menyiapkan integrasi metode.

Untuk mengatur integrasi metode

1. Pilih sumber daya/{operand1}/{operand2}/{operator}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Lambda.
4. Matikan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda dan masukkan. **Calc**
6. Tetap aktifkan batas waktu default.
7. Pilih metode Buat.

Anda sekarang membuat template pemetaan untuk memetakan tiga parameter jalur URL, dideklarasikan ketika sumber daya /calc/ {operand1}/{operand2}/{operator} dibuat, ke dalam nilai properti yang ditunjuk di objek JSON. Karena jalur URL harus dikodekan URL, operator divisi harus ditentukan sebagai pengganti. %2F / Template ini menerjemahkan %2F ke dalam ' / ' sebelum meneruskannya ke fungsi Lambda.

Untuk membuat template pemetaan

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih template Pemetaan.
4. Untuk jenis Konten, masukkan **application/json**.
5. Untuk badan Template, masukkan kode berikut:

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

6. Pilih Simpan.

Anda sekarang dapat menguji GET metode Anda untuk memverifikasi bahwa itu telah diatur dengan benar untuk menjalankan fungsi Lambda dan meneruskan output asli melalui respons integrasi tanpa pemetaan.

Untuk menguji **GET** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk Path, lakukan hal berikut:
 - a. Untuk operand1, masukkan. **1**
 - b. Untuk operand2, masukkan. **1**
 - c. Untuk operator, masukkan **+**.
3. Pilih Uji.
4. Hasilnya akan terlihat seperti ini:

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

Selanjutnya, Anda memodelkan struktur data payload respons metode setelah `result` skema.

Secara default, badan respons metode diberi model kosong. Ini akan menyebabkan badan respons integrasi dilewatkan tanpa pemetaan. Namun, ketika Anda membuat SDK untuk salah satu bahasa tipe kuat, seperti Java atau Objective-C, pengguna SDK Anda akan menerima objek kosong sebagai hasilnya. Untuk memastikan bahwa klien REST dan klien SDK menerima hasil yang diinginkan, Anda harus memodelkan data respons menggunakan skema yang telah ditentukan. Di sini Anda akan mendefinisikan model untuk badan respons metode dan untuk membuat template pemetaan untuk menerjemahkan badan respons integrasi ke dalam badan respons metode.

Untuk membuat respons metode

1. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.
2. Di bawah Badan respons, pilih Tambahkan model.
3. Untuk jenis Konten, masukkan **application/json**.
4. Untuk Model, pilih hasil.

5. Pilih Simpan.

Menyetel model untuk badan respons metode memastikan bahwa data respons akan dilemparkan ke `result` objek SDK tertentu. Untuk memastikan bahwa data respons integrasi dipetakan sesuai, Anda memerlukan template pemetaan.

Untuk membuat template pemetaan

1. Pada tab Respons integrasi, di bawah Default - Respons, pilih Edit.
2. Pilih template Pemetaan.
3. Untuk jenis Konten, masukkan **application/json**.
4. Untuk badan Template, masukkan kode berikut:

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

5. Pilih Simpan.

Untuk menguji template pemetaan

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk Path, lakukan hal berikut:
 - a. Untuk operand1, masukkan. **1**
 - b. Untuk operand2, masukkan. **2**
 - c. Untuk operator, masukkan **+**.
3. Pilih Uji.
4. Hasilnya akan terlihat seperti berikut:

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

Pada titik ini, Anda hanya dapat memanggil API menggunakan fitur Uji di konsol API Gateway. Untuk membuatnya tersedia bagi klien, Anda harus menerapkan API Anda. Selalu pastikan untuk menerapkan ulang API Anda setiap kali Anda menambahkan, memodifikasi, atau menghapus sumber daya atau metode, memperbarui pemetaan data, atau memperbarui pengaturan tahap. Jika tidak, fitur atau pembaruan baru tidak akan tersedia untuk klien API Anda. sebagai berikut:

Untuk menerapkan API

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **Prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.
6. (Opsional) Di bawah detail Tahap, untuk Invoke URL, Anda dapat memilih ikon salin untuk menyalin URL pemanggilan API Anda. Anda dapat menggunakan ini dengan alat seperti [Postman](#) dan [cURL](#) untuk menguji API Anda.

Note

Selalu terapkan ulang API Anda setiap kali Anda menambahkan, memodifikasi, atau menghapus sumber daya atau metode, memperbarui pemetaan data, atau memperbarui pengaturan tahap. Jika tidak, fitur atau pembaruan baru tidak akan tersedia untuk klien API Anda.

Definisi OpenAPI dari contoh API terintegrasi dengan fungsi Lambda

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\" : \"$res.b\", \n  \"op\" : \"$res.op\", \n  \"c\" : \"$res.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_match",
    "httpMethod": "POST",
  }
}

```

```

      "requestTemplates": {
        "application/json": "{\n  \"a\": \"${input.params('operand1')}\",\n  \"b\": \"${input.params('operand2')}\",\n  \"op\": \"${input.params('operator')}\",\n  \"c\": \"${input.params('operand3')}\"\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\" : $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')\"/\n#{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: Buat REST API sebagai proxy Amazon S3

Sebagai contoh untuk menampilkan menggunakan REST API di API Gateway ke proxy Amazon S3, bagian ini menjelaskan cara membuat dan mengonfigurasi REST API untuk mengekspos operasi Amazon S3 berikut:

- Paparkan GET di sumber daya root API untuk [mencantumkan semua bucket Amazon S3](#) pemanggil.
- Paparkan GET pada resource Folder untuk [melihat daftar semua objek di bucket Amazon S3](#).
- Paparkan GET pada Folder/Item sumber daya untuk [melihat atau mengunduh objek dari bucket Amazon S3](#).

Anda mungkin ingin mengimpor API sampel sebagai proxy Amazon S3, seperti yang ditunjukkan pada. [Definisi OpenAPI dari API sampel sebagai proxy Amazon S3](#) Sampel ini berisi metode yang lebih terbuka. Untuk petunjuk tentang cara mengimpor API menggunakan definisi OpenAPI, lihat. [Kembangkan REST API menggunakan OpenAPI di API Gateway](#)

Note

Untuk mengintegrasikan API Gateway API Anda dengan Amazon S3, Anda harus memilih wilayah tempat layanan API Gateway dan Amazon S3 tersedia. Untuk ketersediaan wilayah, lihat [Titik Akhir dan Kuota Amazon API Gateway](#).

Topik

- [Siapkan izin IAM untuk API untuk menjalankan tindakan Amazon S3](#)
- [Buat sumber daya API untuk mewakili sumber daya Amazon S3](#)
- [Paparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil](#)
- [Mengekspos metode API untuk mengakses bucket Amazon S3](#)
- [Mengekspos metode API untuk mengakses objek Amazon S3 dalam bucket](#)
- [Definisi OpenAPI dari API sampel sebagai proxy Amazon S3](#)
- [Panggil API menggunakan klien REST API](#)

Siapkan izin IAM untuk API untuk menjalankan tindakan Amazon S3

Untuk mengizinkan API menjalankan tindakan Amazon S3, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM. Pada langkah ini, Anda membuat peran IAM baru.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran **APIGatewayS3ProxyPolicy**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.
8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **AmazonS3FullAccess** dan pilih Tambahkan izin.

Note

Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Buat sumber daya API untuk mewakili sumber daya Amazon S3

Anda menggunakan sumber daya root (/) API sebagai wadah bucket Amazon S3 pemanggil yang diautentikasi. Anda juga membuat Folder dan Item sumber daya untuk mewakili bucket Amazon S3 tertentu dan objek Amazon S3 tertentu, masing-masing. Nama folder dan kunci objek akan ditentukan, dalam bentuk parameter jalur sebagai bagian dari URL permintaan, oleh pemanggil.

Note

Saat mengakses objek yang kunci objeknya termasuk / atau karakter khusus lainnya, karakter harus dikodekan URL. Misalnya, `test/test.txt` harus dikodekan ke `test %2Ftest.txt`

Untuk membuat sumber daya API yang mengekspos fitur layanan Amazon S3

1. Saat Wilayah AWS Anda membuat bucket Amazon S3, buat API bernama myS3. Sumber daya root API ini (/) mewakili layanan Amazon S3. Pada langkah ini, Anda membuat dua sumber daya tambahan/{folder} dan/{item}.
2. Pilih Buat sumber daya.
3. Matikan sumber daya Proxy.
4. Untuk jalur Sumber Daya, pilih/.
5. Untuk Nama sumber daya, masukkan **{folder}**.
6. Biarkan CORS (Cross Origin Resource Sharing) tidak dicentang.
7. Pilih Buat sumber daya.
8. Pilih sumber daya/{folder}, lalu pilih Buat sumber daya.
9. Gunakan langkah-langkah sebelumnya untuk membuat sumber daya turunan dari/{folder} bernama{item}.

API final Anda akan terlihat mirip dengan yang berikut ini:

Resources

API actions ▼ Deploy API

Create resource

- /
- {folder}
- {item}**

Resource details Delete Update documentation Enable CORS

Path: /{folder}/{item} Resource ID: efg456

Methods (0) Delete Create method

Method type	Integration type	Authorization	API key
No methods			
No methods defined.			

Paparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil

Mendapatkan daftar bucket Amazon S3 dari penelepon melibatkan pemanggilan tindakan [GET Service di](#) Amazon S3. Pada sumber daya root API, (/), buat metode GET. Konfigurasi metode GET untuk diintegrasikan dengan Amazon S3, sebagai berikut.

Untuk membuat dan menginisialisasi metode API **GET /**

1. Pilih sumber daya/, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.
5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override.

Dengan penggantian jalur, API Gateway meneruskan permintaan klien ke Amazon S3 sebagai permintaan gaya jalur API [Amazon S3 REST API yang sesuai, di mana sumber daya](#) Amazon S3 diekspresikan oleh jalur sumber daya pola. s3-host-name/bucket/key API Gateway menyetel s3-host-name dan meneruskan klien yang ditentukan bucket dan key dari klien ke Amazon S3.

9. Untuk Path override, masukkan/.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **APIGatewayS3ProxyPolicy**
11. Pilih Pengaturan permintaan Metode.

Anda menggunakan pengaturan permintaan metode untuk mengontrol siapa yang dapat memanggil metode API Anda ini.

12. Untuk Otorisasi, dari menu tarik-turun, pilih. AWS_IAM

Method request settings

Authorization

None

None

AWS IAM

☐ API key required

Operation name - optional

GetPets

13. Pilih metode Buat.

Pengaturan ini mengintegrasikan GET `https://your-api-host/stage/` permintaan frontend dengan backend. GET `https://your-s3-host/`

Agar API Anda mengembalikan respons dan pengecualian yang berhasil dengan benar ke penelepon, Anda mendeklarasikan respons 200, 400, dan 500 dalam respons Metode. Anda menggunakan pemetaan default untuk 200 tanggapan sehingga tanggapan backend dari kode status yang tidak dideklarasikan di sini akan dikembalikan ke pemanggil sebagai 200 respons.

Untuk mendeklarasikan tipe respons untuk metode **GET** /

1. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.
2. Pilih Add header dan lakukan hal berikut:
 - a. Untuk nama Header, masukkan **Content-Type**.
 - b. Pilih Tambahkan header.

Ulangi langkah-langkah ini untuk membuat **Timestamp** header dan **Content-Length** header.

3. Pilih Simpan.
4. Pada tab Respons metode, di bawah Respons metode, pilih Buat respons.
5. Untuk kode status HTTP, masukkan 400.

Anda tidak menetapkan header apa pun untuk respons ini.

6. Pilih Simpan.
7. Ulangi langkah-langkah berikut untuk membuat respons 500.

Anda tidak menetapkan header apa pun untuk respons ini.

Karena respons integrasi yang berhasil dari Amazon S3 mengembalikan bucket list sebagai payload XHTML dan respons metode default dari API Gateway mengembalikan payload JSON, Anda harus menetapkan nilai parameter header Content-Type backend ke counterpart frontend. Jika tidak, klien akan menerima `application/json` untuk tipe konten ketika badan respons sebenarnya adalah string XHTML. Prosedur berikut menunjukkan cara mengatur ini. Selain itu, Anda juga ingin menampilkan parameter header lainnya ke klien, seperti Tanggal dan Panjang Konten.

Untuk mengatur pemetaan header respons untuk metode GET/

1. Pada tab Respons integrasi, di bawah Default - Respons, pilih Edit.
2. Untuk header Content-Length, masukkan **`integration.response.header.Content-Length`** untuk nilai pemetaan.
3. Untuk header Content-Type, masukkan **`integration.response.header.Content-Type`** untuk nilai pemetaan.
4. Untuk header Timestamp, masukkan **`integration.response.header.Date`** untuk nilai pemetaan.
5. Pilih Simpan. Hasilnya akan terlihat mirip dengan yang berikut:

[Method request](#)
[Integration request](#)
[Integration response](#)
[Met](#)

Integration responses

[Create response](#)

Default - Response

[Edit](#)
[Delete](#)

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.

6. Pada tab Respons Integrasi, di bawah Respons integrasi, pilih Buat respons.
7. Untuk regex status HTTP, masukkan. `4\d{2}` Ini memetakan semua kode status respons HTTP 4xx ke respons metode.
8. Untuk kode status respons Metode, pilih `400`.
9. Pilih Buat.
10. Ulangi langkah-langkah berikut untuk membuat respons integrasi untuk respons metode 500. Untuk regex status HTTP, masukkan. `5\d{2}`

Sebagai praktik yang baik, Anda dapat menguji API yang telah Anda konfigurasi sejauh ini.

Untuk menguji **GET** / metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Pilih Uji. Hasilnya akan terlihat seperti gambar berikut:

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-
```

Mengekspos metode API untuk mengakses bucket Amazon S3

Untuk bekerja dengan bucket Amazon S3, Anda mengekspos GET metode pada resource/{folder} untuk mencantumkan objek dalam bucket. Instruksi mirip dengan yang dijelaskan dalam [Paparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil](#). Untuk metode lainnya, Anda dapat mengimpor contoh API di sini, [Definisi OpenAPI dari API sampel sebagai proxy Amazon S3](#).

Untuk mengekspos metode GET pada sumber daya folder

1. Pilih sumber daya/{folder}, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.
5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override.
9. Untuk Path override, masukkan **{bucket}**.
10. Untuk peran Eksekusi, masukkan peran ARN untuk **APIGatewayS3ProxyPolicy**
11. Pilih metode Buat.

Anda mengatur parameter {folder} jalur di URL titik akhir Amazon S3. Anda perlu memetakan parameter {folder} jalur permintaan metode ke parameter {bucket} jalur permintaan integrasi.

Untuk memetakan **{folder}** ke **{bucket}**

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL, lalu pilih Tambahkan parameter jalur.
3. Untuk Nama, masukkan **bucket**.
4. Untuk Dipetakan dari, masukkan **method.request.path.folder**.
5. Pilih Simpan.

Sekarang, Anda menguji API Anda.

Untuk menguji **/{{folder}} GET** metode.

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path, untuk folder, masukkan nama bucket Anda.
3. Pilih Uji.

Hasil tes akan berisi daftar objek di bucket Anda.

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

ⓘ **/{folder} - GET method test results**
Request

Latency ms

Status

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-
```

Mengekspos metode API untuk mengakses objek Amazon S3 dalam bucket

Amazon S3 mendukung tindakan GET, DELETE, HEAD, OPTIONS, POST, dan PUT untuk mengakses dan mengelola objek dalam bucket tertentu. Dalam tutorial ini, Anda mengekspos GET metode pada {folder}/{item} sumber daya untuk mendapatkan gambar dari ember. Untuk aplikasi {folder}/{item} sumber daya lainnya, lihat contoh API, [Definisi OpenAPI dari API sampel sebagai proxy Amazon S3](#).

Untuk mengekspos metode GET pada sumber daya item

1. Pilih sumber daya/{item}, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.

5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override.
9. Untuk penggantian Path, masukkan `{bucket}/{object}`.
10. Untuk peran Eksekusi, masukkan peran ARN untuk **APIGatewayS3ProxyPolicy**
11. Pilih metode Buat.

Anda mengatur parameter `{folder}` dan `{item}` jalur di URL titik akhir Amazon S3. Anda perlu memetakan parameter jalur permintaan metode ke parameter jalur permintaan integrasi.

Dalam langkah ini, Anda melakukan hal berikut:

- Petakan parameter `{folder}` jalur permintaan metode ke parameter `{bucket}` jalur permintaan integrasi.
- Petakan parameter `{item}` jalur permintaan metode ke parameter `{object}` jalur permintaan integrasi.

Untuk memetakan **`{folder}`** ke **`{bucket}`** dan **`{item}`** ke **`{object}`**

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL.
3. Pilih Tambahkan parameter jalur.
4. Untuk Nama, masukkan **bucket**.
5. Untuk Dipetakan dari, masukkan **`method.request.path.folder`**.
6. Pilih Tambahkan parameter jalur.
7. Untuk Nama, masukkan **object**.
8. Untuk Dipetakan dari, masukkan **`method.request.path.item`**.
9. Pilih Simpan.

Untuk menguji **`/{folder}/{object}` GET** metode.

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path, untuk folder, masukkan nama bucket Anda.

3. Di bawah Path, untuk item, masukkan nama item.
4. Pilih Uji.

Badan respons akan berisi isi item.

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		

Permintaan mengembalikan teks biasa (“Hello world”) dengan benar sebagai konten file yang ditentukan (test.txt) di bucket Amazon S3 yang diberikan (amzn-s3-demo-bucket).

Untuk mengunduh atau mengunggah file biner, yang di API Gateway dianggap sebagai hal apa pun selain konten JSON yang dikodekan utf-8, diperlukan pengaturan API tambahan. Ini diuraikan sebagai berikut:

Untuk mengunduh atau mengunggah file biner dari S3

1. Daftarkan jenis media dari file yang terpengaruh ke API `binaryMediaTypes`. Anda dapat melakukan ini di konsol:
 - a. Pilih pengaturan API untuk API.
 - b. Di bawah Jenis media biner, pilih Kelola jenis media.
 - c. Pilih Tambahkan jenis media biner, lalu masukkan jenis media yang diperlukan, misalnya, `image/png`.
 - d. Pilih Simpan perubahan untuk menyimpan pengaturan.
2. Tambahkan header `Content-Type` (untuk diunggah) and/or `Accept` (untuk diunduh) ke permintaan metode untuk meminta klien menentukan jenis media biner yang diperlukan dan memetakannya ke permintaan integrasi.
3. Setel Penanganan Konten ke `Passthrough` dalam permintaan integrasi (untuk diunggah) dan dalam respons integrasi (untuk diunduh). Pastikan tidak ada template pemetaan yang ditentukan untuk jenis konten yang terpengaruh. Untuk informasi selengkapnya, lihat [the section called "Transformasi data"](#).

Batas ukuran muatan adalah 10 MB. Lihat [Kuota untuk mengonfigurasi dan menjalankan REST API di API Gateway](#).

Pastikan file di Amazon S3 memiliki jenis konten yang benar yang ditambahkan sebagai metadata file. Untuk konten media yang dapat disederhanakan, `Content-Disposition:inline` mungkin juga perlu ditambahkan ke metadata.

Untuk informasi selengkapnya tentang dukungan biner di API Gateway, lihat [Konversi jenis konten di API Gateway](#).

Definisi OpenAPI dari API sampel sebagai proxy Amazon S3

Definisi OpenAPI berikut menjelaskan API yang berfungsi sebagai proxy Amazon S3. API ini berisi lebih banyak operasi Amazon S3 daripada API yang Anda buat dalam tutorial. Metode berikut diekspos dalam definisi OpenAPI:

- Paparkan GET di sumber daya root API untuk [mencantumkan semua bucket Amazon S3](#) pemanggil.
- Paparkan GET pada resource Folder untuk [melihat daftar semua objek di bucket Amazon S3](#).

- Paparkan PUT pada sumber daya Folder untuk [menambahkan bucket ke Amazon S3](#).
- Paparkan DELETE pada resource Folder untuk [menghapus bucket dari Amazon S3](#).
- Paparkan GET pada Folder/Item sumber daya untuk [melihat atau mengunduh objek dari bucket Amazon S3](#).
- Paparkan PUT pada Folder/Item sumber daya untuk [mengunggah objek ke bucket Amazon S3](#).
- Paparkan HEAD pada Folder/Item sumber daya untuk [mendapatkan metadata objek di bucket Amazon S3](#).
- Paparkan DELETE pada Folder/Item sumber daya untuk [menghapus objek dari bucket Amazon S3](#).

Untuk petunjuk tentang cara mengimpor API menggunakan definisi OpenAPI, lihat. [Kembangkan REST API menggunakan OpenAPI di API Gateway](#)

Untuk petunjuk tentang cara membuat API serupa, lihat [Tutorial: Buat REST API sebagai proxy Amazon S3](#).

Untuk mempelajari cara menjalankan API ini menggunakan [Postman](#), yang mendukung otorisasi AWS IAM, lihat. [Panggil API menggunakan klien REST API](#)

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"
        }
    }
}
}
}
}

```

Panggil API menggunakan klien REST API

Untuk memberikan end-to-end tutorial, kami sekarang menunjukkan cara memanggil API menggunakan [Postman](#), yang mendukung otorisasi AWS IAM.

Untuk memanggil API proxy Amazon S3 kami menggunakan Postman

1. Menerapkan atau menerapkan ulang API. Catat URL dasar API yang ditampilkan di sebelah Invoke URL di bagian atas Stage Editor.
2. Luncurkan Postman.
3. Pilih Otorisasi dan kemudian pilih AWS Signature. Masukkan ID Kunci Akses dan Kunci Akses Rahasia pengguna IAM Anda ke dalam bidang AccessKey dan SecretKey input, masing-masing.

Masukkan Wilayah AWS ke mana API Anda digunakan di kotak teks AWS Region. Masukkan `execute-api` di kolom input Nama Layanan.

Anda dapat membuat sepasang kunci dari tab Security Credentials dari akun pengguna IAM Anda di IAM Management Console.

4. Untuk menambahkan bucket yang diberi nama `amzn-s3-demo-bucket` ke akun Amazon S3 Anda di wilayah ini `{region}`:

- a. Pilih PUT dari daftar metode drop-down dan ketik URL metode (`https://api-id.execute-api.aws-region.amazonaws.com/stage/folder-name`
- b. Tetapkan nilai Content-Type header sebagai `application/xml`. Anda mungkin perlu menghapus header yang ada sebelum menyetel jenis konten.
- c. Pilih item menu Body dan ketik fragmen XHTML berikut sebagai badan permintaan:

```
<CreateBucketConfiguration>
  <LocationConstraint>{region}</LocationConstraint>
</CreateBucketConfiguration>
```

- d. Pilih Kirim untuk mengirimkan permintaan. Jika berhasil, Anda harus menerima 200 OK respons dengan muatan kosong.
5. Untuk menambahkan file teks ke ember, ikuti petunjuk di atas. Jika Anda menentukan nama bucket **amzn-s3-demo-bucket** for `{folder}` dan nama file **Readme.txt** for `{item}` di URL dan memberikan string teks **Hello, World!** sebagai isi file (sehingga menjadikannya payload permintaan), permintaan menjadi

```
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!
```

Jika semuanya berjalan dengan baik, Anda harus menerima 200 OK respons dengan muatan kosong.

6. Untuk mendapatkan konten Readme.txt file yang baru saja kita tambahkan ke amzn-s3-demo-bucket bucket, lakukan permintaan GET seperti berikut ini:

```
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
```

Jika berhasil, Anda harus menerima 200 OK respons dengan string Hello, World! teks sebagai payload.

7. Untuk mencantumkan item di amzn-s3-demo-bucket bucket, kirimkan permintaan berikut:

```
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
```

Jika berhasil, Anda harus menerima 200 OK respons dengan payload XHTML yang menampilkan satu item dalam bucket yang ditentukan, kecuali jika Anda menambahkan lebih banyak file ke bucket sebelum mengirimkan permintaan ini.

```
<?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

Untuk mengunggah atau mengunduh gambar, Anda perlu mengatur penanganan konten ke CONVERT_TO_BINARY.

Tutorial: Buat REST API sebagai proxy Amazon Kinesis

Halaman ini menjelaskan cara membuat dan mengonfigurasi REST API dengan integrasi AWS tipe untuk mengakses Kinesis.

Note

Untuk mengintegrasikan API Gateway API Anda dengan Kinesis, Anda harus memilih wilayah tempat layanan API Gateway dan Kinesis tersedia. Untuk ketersediaan wilayah, lihat [Titik Akhir Layanan dan Kuota](#).

Untuk tujuan ilustrasi, kami membuat contoh API untuk memungkinkan klien melakukan hal berikut:

1. Daftar aliran yang tersedia pengguna di Kinesis
2. Membuat, mendeskripsikan, atau menghapus aliran tertentu
3. Membaca catatan data dari atau menulis catatan data ke dalam aliran yang ditentukan

Untuk menyelesaikan tugas-tugas sebelumnya, API mengekspos metode pada berbagai sumber daya untuk menjalankan yang berikut ini, masing-masing:

1. ListStreamsTindakan dalam Kinesis

2. , `CreateStreamDescribeStream`, atau `DeleteStream` tindakan
3. Tindakan `GetRecords` atau `PutRecords` (termasuk `PutRecord`) dalam Kinesis

Secara khusus, kami membangun API sebagai berikut:

- Paparkan metode HTTP GET pada `/streams` sumber daya API dan integrasikan metode tersebut dengan [ListStreams](#) tindakan di Kinesis untuk mencantumkan aliran di akun pemanggil.
- Paparkan metode HTTP POST pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan [CreateStream](#) tindakan di Kinesis untuk membuat aliran bernama di akun pemanggil.
- Paparkan metode HTTP GET pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan [DescribeStream](#) tindakan di Kinesis untuk mendeskripsikan aliran bernama di akun pemanggil.
- Paparkan metode HTTP DELETE pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan [DeleteStream](#) tindakan di Kinesis untuk menghapus aliran di akun pemanggil.
- Paparkan metode HTTP PUT pada `/streams/{stream-name}/record` sumber daya API dan integrasikan metode dengan [PutRecord](#) tindakan di Kinesis. Ini memungkinkan klien untuk menambahkan catatan data tunggal ke aliran bernama.
- Paparkan metode HTTP PUT pada `/streams/{stream-name}/records` sumber daya API dan integrasikan metode dengan [PutRecords](#) tindakan di Kinesis. Ini memungkinkan klien untuk menambahkan daftar catatan data ke aliran bernama.
- Paparkan metode HTTP GET pada `/streams/{stream-name}/records` sumber daya API dan integrasikan metode dengan [GetRecords](#) tindakan di Kinesis. Hal ini memungkinkan klien untuk daftar catatan data dalam aliran bernama, dengan iterator shard tertentu. Sebuah iterator shard menentukan posisi shard dari mana untuk mulai membaca catatan data secara berurutan.
- Paparkan metode HTTP GET pada `/streams/{stream-name}/sharditerator` sumber daya API dan integrasikan metode dengan [GetShardIterator](#) tindakan di Kinesis. Metode pembantu ini harus diberikan pada `ListStreams` tindakan di Kinesis.

Anda dapat menerapkan instruksi yang disajikan di sini untuk tindakan Kinesis lainnya. Untuk daftar lengkap tindakan Kinesis, lihat Referensi API [Amazon Kinesis](#).

Alih-alih menggunakan konsol API Gateway untuk membuat API sampel, Anda dapat mengimpor API sampel ke API Gateway menggunakan API Gateway [Import API](#). Untuk informasi tentang cara menggunakan API Impor, lihat [Kembangkan REST API menggunakan OpenAPI di API Gateway](#).

Membuat peran dan kebijakan IAM untuk API untuk mengakses Kinesis

Untuk mengizinkan API menjalankan tindakan Kinesis, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM. Pada langkah ini, Anda membuat peran IAM baru.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran **APIGatewayKinesisProxyPolicy**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.
8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **AmazonKinesisFullAccess** dan pilih Tambahkan izin.

Note

Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Buat API sebagai proxy Kinesis

Gunakan langkah-langkah berikut untuk membuat API di konsol API Gateway.

Untuk membuat API sebagai proxy AWS layanan untuk Kinesis

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat muncul Create Example API muncul, pilih OK.

Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.

3. Pilih API Baru.
4. Dalam nama API, masukkan **KinesisProxy**. Simpan nilai default untuk semua bidang lainnya.
5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Untuk jenis alamat IP, pilih IPv4.
7. Pilih Buat API.

Setelah API dibuat, konsol API Gateway menampilkan halaman Resources, yang hanya berisi sumber daya root (/) API.

Daftar aliran dalam Kinesis

Kinesis mendukung ListStreams tindakan dengan panggilan REST API berikut:

```
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>

{
  ...
}
```

Dalam permintaan REST API di atas, tindakan ditentukan dalam parameter Action kueri. Atau, Anda dapat menentukan tindakan di X-Amz-Target header, sebagai gantinya:

```
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
{
  ...
}
```

Dalam tutorial ini, kita menggunakan parameter query untuk menentukan tindakan.

Untuk mengekspos tindakan Kinesis di API, tambahkan `/streams` resource ke root API. Kemudian atur GET metode pada sumber daya dan integrasikan metode dengan `ListStreams` aksi Kinesis.

Prosedur berikut menjelaskan cara membuat daftar aliran Kinesis menggunakan konsol API Gateway.

Untuk membuat daftar aliran Kinesis dengan menggunakan konsol API Gateway

1. Pilih `/` sumber daya, lalu pilih Buat sumber daya.
2. Untuk Nama sumber daya, masukkan **streams**.
3. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
4. Pilih Buat sumber daya.
5. Pilih `/streams` sumber daya, lalu pilih Create method, lalu lakukan hal berikut:
 - a. Untuk tipe Metode, pilih GET.

 Note

Kata kerja HTTP untuk metode yang dipanggil oleh klien mungkin berbeda dari kata kerja HTTP untuk integrasi yang diperlukan oleh backend. Kami memilih GET di sini, karena aliran daftar secara intuitif merupakan operasi BACA.

- b. Untuk jenis Integrasi, pilih AWS layanan.
- c. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
- d. Untuk Layanan AWS, pilih Kinesis.
- e. Biarkan AWS subdomain kosong.

- f. Untuk metode HTTP, pilih POST.

 Note

Kami memilih POST di sini karena Kinesis mengharuskan `ListStreams` tindakan dipanggil dengannya.

- g. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
 - h. Untuk nama Action, masukkan **ListStreams**.
 - i. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
 - j. Pertahankan default Passthrough untuk Penanganan Konten.
 - k. Pilih metode Buat.
6. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
 7. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
 8. Pilih parameter header permintaan URL, lalu lakukan hal berikut:
 - a. Pilih Tambahkan parameter header permintaan.
 - b. Untuk Nama, masukkan **Content-Type**.
 - c. Untuk Dipetakan dari, masukkan '**application/x-amz-json-1.1**'.

Kami menggunakan pemetaan parameter permintaan untuk mengatur Content-Type header ke nilai statis '`application/x-amz-json-1.1`' untuk menginformasikan Kinesis bahwa input adalah versi tertentu dari JSON.

9. Pilih Mapping templates, lalu pilih Add mapping template, dan lakukan hal berikut:
 - a. Untuk Content-Type, masukkan. **application/json**
 - b. Untuk badan Template, masukkan **{}**.
 - c. Pilih Simpan.

[ListStreams](#)Permintaan mengambil muatan dari format JSON berikut:

```
{
  "ExclusiveStartStreamName": "string",
  "Limit": number
}
```

```
}
```

Namun, propertinya opsional. Untuk menggunakan nilai default, kami memilih payload JSON kosong di sini.

10. Uji metode GET pada sumber daya `/streams` untuk menjalankan tindakan `ListStreams` di Kinesis:

Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

Pilih Tes untuk menguji metode Anda.

Jika Anda telah membuat dua aliran bernama “MyStream” dan “YourStream” di Kinesis, pengujian yang berhasil mengembalikan respons 200 OK yang berisi muatan berikut:

```
{
  "HasMoreStreams": false,
  "StreamNames": [
    "myStream",
    "yourStream"
  ]
}
```

Buat, jelaskan, dan hapus aliran di Kinesis

Membuat, mendeskripsikan, dan menghapus aliran di Kinesis melibatkan pembuatan permintaan Kinesis REST API berikut, masing-masing:

```
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"
}
```

Kita dapat membangun API untuk menerima input yang diperlukan sebagai payload JSON dari permintaan metode dan meneruskan payload ke permintaan integrasi. Namun, untuk memberikan lebih banyak contoh pemetaan data antara permintaan metode dan integrasi, serta respons metode dan integrasi, kami membuat API kami agak berbeda.

Kami mengekspos GET, POST, dan metode Delete HTTP pada to-be-named Stream sumber daya. Kami menggunakan variabel {stream-name} jalur sebagai pengganti sumber daya aliran dan mengintegrasikan metode API ini dengan Kinesis, dan tindakan DescribeStream, CreateStream, dan DeleteStream. Kami mengharuskan klien meneruskan data input lain sebagai header, parameter kueri, atau muatan permintaan metode. Kami menyediakan template pemetaan untuk mengubah data ke payload permintaan integrasi yang diperlukan.

Untuk membuat sumber daya {stream-name}

1. Pilih sumber daya /streams, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Untuk jalur Sumber Daya, pilih /streams.
4. Untuk Nama sumber daya, masukkan **{stream-name}**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Untuk mengonfigurasi dan menguji metode GET pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **DescribeStream**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk GET /streams metode tersebut.

14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan GET /streams/{stream-name} metode ke permintaan POST /?Action=DescribeStream integrasi:

```
{
  "StreamName": "$input.params('stream-name')"
}
```

Template pemetaan ini menghasilkan payload permintaan integrasi yang diperlukan untuk `DescribeStream` tindakan Kinesis dari nilai parameter jalur permintaan `stream-name` metode.

15. Untuk menguji `GET /stream/{stream-name}` metode untuk menjalankan `DescribeStream` tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis yang ada.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan dengan muatan yang mirip dengan berikut ini:

```
{
  "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"  
}  
}
```

Setelah menerapkan API, Anda dapat membuat permintaan REST terhadap metode API ini:

```
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
```

Untuk mengkonfigurasi dan menguji metode POST pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.
2. Untuk jenis Metode, pilih POST.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **CreateStream**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk GET /streams metode tersebut.

14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan POST /streams/{stream-name} metode ke permintaan POST /?Action=CreateStream integrasi:

```
{
  "ShardCount": #if($input.path('$.ShardCount') == '') 5 #else
    $input.path('$.ShardCount') #end,
  "StreamName": "$input.params('stream-name')"
}
```

Dalam template pemetaan sebelumnya, kami menetapkan ShardCount ke nilai tetap 5 jika klien tidak menentukan nilai dalam payload permintaan metode.

15. Untuk menguji POST /stream/{stream-name} metode untuk menjalankan CreateStream tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis baru.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan tanpa data.

Setelah menerapkan API, Anda juga dapat membuat permintaan REST API terhadap metode POST pada resource Stream untuk menjalankan CreateStream tindakan di Kinesis:

```
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
}
```

Konfigurasi dan uji metode DELETE pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.

2. Untuk jenis Metode, pilih DELETE.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **DeleteStream**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk GET /streams metode tersebut.

14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan DELETE /streams/{stream-name} metode ke permintaan integrasi yang sesuai dari POST /?Action=DeleteStream:

```
{
  "StreamName": "$input.params('stream-name')"
}
```

Template pemetaan ini menghasilkan input yang diperlukan untuk DELETE /streams/{stream-name} tindakan dari nama jalur URL yang disediakan klien. stream-name

15. Untuk menguji DELETE /stream/{stream-name} metode untuk menjalankan DeleteStream tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis yang ada.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan tanpa data.

Setelah menerapkan API, Anda juga dapat membuat permintaan REST API berikut terhadap metode DELETE pada sumber daya Stream untuk memanggil DeleteStream tindakan di Kinesis:

```
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

{}
```

Dapatkan catatan dari dan tambahkan catatan ke aliran di Kinesis

Setelah Anda membuat aliran di Kinesis, Anda dapat menambahkan catatan data ke aliran dan membaca data dari aliran. Menambahkan catatan data melibatkan pemanggilan [PutRecords](#) atau [PutRecord](#) tindakan dalam Kinesis. Yang pertama menambahkan beberapa catatan sedangkan yang terakhir menambahkan satu catatan ke aliran.

```
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"
}
```

atau

```
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"
}
```

Di sini, `StreamName` mengidentifikasi aliran target untuk menambahkan catatan. `StreamName`, `Data`, dan `PartitionKey` diperlukan data input. Dalam contoh kita, kita menggunakan nilai default untuk semua data input opsional dan tidak akan secara eksplisit menentukan nilai untuk mereka dalam input ke permintaan metode.

Membaca data dalam Kinesis sama dengan memanggil tindakan: [GetRecords](#)

```
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
}
```


Di sini, aliran sumber dari mana kita mendapatkan catatan ditentukan dalam `ShardIterator` nilai yang diperlukan, seperti yang ditunjukkan dalam tindakan Kinesis berikut untuk mendapatkan iterator pecahan:

```
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"
}
```

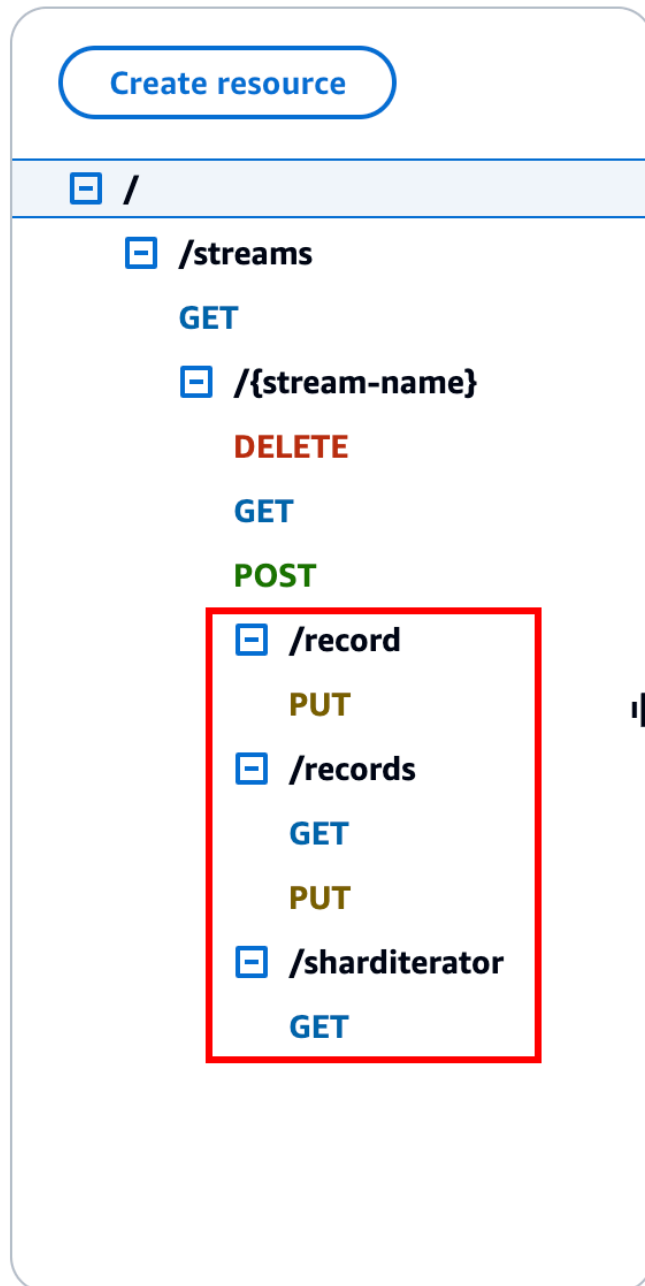
Untuk `PutRecords` tindakan `GetRecords` dan, kami mengekspos PUT metode GET dan, masing-masing, pada `/records` sumber daya yang ditambahkan ke sumber daya aliran bernama `()/{stream-name}`. Demikian pula, kami mengekspos `PutRecord` tindakan sebagai PUT metode pada `/record` sumber daya.

Karena `GetRecords` tindakan mengambil sebagai input `ShardIterator` nilai, yang diperoleh dengan memanggil aksi `GetShardIterator` helper, kami mengekspos metode GET helper pada `ShardIterator` resource `() ./sharditerator`

Untuk membuat sumber daya `/record`, `/records`, dan `/sharditerator`

1. Pilih sumber daya `{stream-name}`, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Untuk jalur Sumber Daya, pilih `{stream-name}`.
4. Untuk Nama sumber daya, masukkan **record**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

7. Ulangi langkah sebelumnya untuk membuat sumber `/records` dan `/sharditerator`. API final akan terlihat seperti berikut:



Empat prosedur berikut menjelaskan cara mengatur masing-masing metode, cara memetakan data dari permintaan metode ke permintaan integrasi, dan cara menguji metode.

Untuk mengatur dan menguji **PUT /streams/{stream-name}/record** metode yang akan dipanggil **PutRecord** di Kinesis:

1. Pilih `/record`, lalu pilih `Create method`.
2. Untuk jenis Metode, pilih `PUT`.
3. Untuk jenis Integrasi, pilih `AWS layanan`.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih `Kinesis`.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih `POST`.
8. Untuk tipe Tindakan, pilih `Gunakan nama tindakan`.
9. Untuk nama Action, masukkan **PutRecord**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default `Passthrough` untuk `Penanganan Konten`.
12. Pilih metode `Buat`.
13. Di bagian `Permintaan integrasi`, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.

14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan `PUT /streams/{stream-name}/record` metode ke permintaan integrasi yang sesuai dari `POST /?Action=PutRecord`:

```
{
  "StreamName": "$input.params('stream-name')",
  "Data": "$util.base64Encode($input.json('$.Data'))",
  "PartitionKey": "$input.path('$.PartitionKey')"
}
```

Template pemetaan ini mengasumsikan bahwa payload permintaan metode adalah dari format berikut:

```
{
```

```
"Data": "some data",  
"PartitionKey": "some key"  
}
```

Data ini dapat dimodelkan dengan skema JSON berikut:

```
{  
  "$schema": "http://json-schema.org/draft-04/schema#",  
  "title": "PutRecord proxy single-record payload",  
  "type": "object",  
  "properties": {  
    "Data": { "type": "string" },  
    "PartitionKey": { "type": "string" }  
  }  
}
```

Anda dapat membuat model untuk menyertakan skema ini dan menggunakan model untuk memfasilitasi pembuatan template pemetaan. Namun, Anda dapat membuat template pemetaan tanpa menggunakan model apa pun.

15. Untuk menguji **PUT /streams/{stream-name}/record** metode, atur variabel `stream-name` jalur ke nama aliran yang ada, berikan muatan format yang diperlukan, lalu kirimkan permintaan metode. Hasil yang berhasil adalah **200 OK** respons dengan muatan format berikut:

```
{  
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",  
  "ShardId": "shardId-0000000000004"  
}
```

Untuk mengatur dan menguji **PUT /streams/{stream-name}/records** metode yang akan dipanggil **PutRecords** di Kinesis

1. Pilih sumber daya `/records`, lalu pilih **Create method**.
2. Untuk jenis Metode, pilih **PUT**.
3. Untuk jenis Integrasi, pilih **AWS layanan**.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.

5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **PutRecords**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk GET `/streams` metode tersebut.

14. Tambahkan template pemetaan berikut untuk memetakan data dari permintaan PUT `/streams/{stream-name}/records` metode ke permintaan integrasi yang sesuai dari POST `/?Action=PutRecords`:

```
{
  "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
}
```

Template pemetaan ini mengasumsikan bahwa payload permintaan metode dapat dimodelkan oleh skema JSON berikut:

```
{
  "$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" }
      }
    }
  }
}
```

Anda dapat membuat model untuk menyertakan skema ini dan menggunakan model untuk memfasilitasi pembuatan template pemetaan. Namun, Anda dapat membuat template pemetaan tanpa menggunakan model apa pun.

Dalam tutorial ini, kami menggunakan dua format payload yang sedikit berbeda untuk menggambarkan bahwa pengembang API dapat memilih untuk mengekspos format data backend ke klien atau menyembunyikannya dari klien. Satu format adalah untuk PUT / streams/{stream-name}/records metode (di atas). Format lain digunakan untuk PUT / streams/{stream-name}/record metode (dalam prosedur sebelumnya). Dalam lingkungan produksi, Anda harus menjaga kedua format tetap konsisten.

15.

Untuk menguji PUT / streams/{stream-name}/records metode, atur variabel stream-name jalur ke aliran yang ada, berikan payload berikut, dan kirimkan permintaan metode.

```
{
  "records": [
    {
      "data": "some data",
      "partition-key": "some key"
    },
    {
      "data": "some other data",
      "partition-key": "some key"
    }
  ]
}
```

Hasil yang berhasil adalah respons 200 OK dengan muatan yang mirip dengan output berikut:

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber": "49559409944537880850133345460167468741933742152373764162",
      "ShardId": "shardId-000000000004"
    },
    {
      "SequenceNumber": "49559409944537880850133345460168677667753356781548470338",
      "ShardId": "shardId-000000000004"
    }
  ]
}
```

Untuk mengatur dan menguji **GET /streams/{stream-name}/sharditerator** metode yang dipanggil **GetShardIterator** di Kinesis

GET /streams/{stream-name}/sharditeratorMetode ini adalah metode pembantu untuk memperoleh iterator shard yang diperlukan sebelum memanggil metode. GET /streams/{stream-name}/records

1. Pilih sumber daya /sharditerator, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **GetShardIterator**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.

12. Pilih parameter string kueri URL.

GetShardIteratorTindakan tersebut membutuhkan masukan dari suatu ShardId nilai. Untuk meneruskan ShardId nilai yang disediakan klien, kami menambahkan parameter **shard-id** kueri ke permintaan metode, seperti yang ditunjukkan pada langkah berikut.

13. Pilih Tambahkan string kueri.

14. Untuk Nama, masukkan **shard-id**.

15. Tetap Diperlukan dan Caching dimatikan.

16. Pilih metode Buat.

17. Di bagian Permintaan integrasi, tambahkan template pemetaan berikut untuk menghasilkan input yang diperlukan (ShardId dan StreamName) ke GetShardIterator tindakan dari shard-id dan stream-name parameter permintaan metode. Selain itu, template pemetaan juga ditetapkan ShardIteratorType TRIM_HORIZON sebagai default.

```
{
  "ShardId": "$input.params('shard-id')",
  "ShardIteratorType": "TRIM_HORIZON",
  "StreamName": "$input.params('stream-name')"
}
```

18. Menggunakan opsi Uji di konsol API Gateway, masukkan nama aliran yang ada sebagai nilai variabel stream-name Path, atur string **shard-id** Kueri ke ShardId nilai yang ada (mis., shard-0000000000004), dan pilih Uji.

Muatan respons yang berhasil mirip dengan output berikut:

```
{
  "ShardIterator": "AAAAAAAAAAFYVN3VlFy..."
}
```

Catat ShardIterator nilainya. Anda membutuhkannya untuk mendapatkan catatan dari aliran.

Untuk mengkonfigurasi dan menguji **GET /streams/{stream-name}/records** metode untuk menjalankan **GetRecords** tindakan di Kinesis

1. Pilih sumber daya /records, lalu pilih Create method.

2. Untuk tipe Metode, pilih GET.

3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan **GetRecords**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih header permintaan HTTP.

GetRecordsTindakan tersebut membutuhkan masukan dari suatu `ShardIterator` nilai.

Untuk meneruskan `ShardIterator` nilai yang disediakan klien, kami menambahkan parameter `Shard-Iterator` header ke permintaan metode.

13. Pilih Tambahkan header.
14. Untuk Nama, masukkan **Shard-Iterator**.
15. Tetap Diperlukan dan Caching dimatikan.
16. Pilih metode Buat.
17. Di bagian Permintaan integrasi, tambahkan template pemetaan badan berikut untuk memetakan nilai parameter `Shard-Iterator` header ke nilai `ShardIterator` properti payload JSON untuk tindakan `GetRecords` di Kinesis.

```
{
  "ShardIterator": "$input.params('Shard-Iterator')"
}
```

18. Menggunakan opsi Uji di konsol API Gateway, masukkan nama aliran yang ada sebagai nilai variabel `stream-name` Path, setel `Shard-Iterator` Header ke `ShardIterator` nilai yang diperoleh dari uji coba GET `/streams/{stream-name}/sharditerator` metode (di atas), dan pilih Uji.

Muatan respons yang berhasil mirip dengan output berikut:

```
{
  "MillisBehindLatest": 0,
  "NextShardIterator": "AAAAAAAAAAAF...",
}
```

```
"Records": [ ... ]  
}
```

Definisi OpenAPI dari API sampel sebagai proxy Kinesis

Berikut ini adalah definisi OpenAPI untuk API sampel sebagai proxy Kinesis yang digunakan dalam tutorial ini.

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  \n  \"StreamName\": \"\${input.params('stream-
name')}\",\n  \n  \"Records\": [\n    {\n      \n      \"Data\":\n      \"\${util.base64Encode($elem.data)}\", \n      \n      \"PartitionKey\":\n      \"\${elem.partition-key}\",\n    }#if($foreach.hasNext),#end\n  ]\n}",
      "application/x-amz-json-1.1": "{\n  \n  \"StreamName\":\n  \"\${input.params('stream-name')}\",\n  \n  \"records\" : [\n    {\n      \n      \"Data\
\n\" : \"\${elem.data}\",\n      \n      \"PartitionKey\" : \"\${elem.partition-key}\",\n
    }#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\":
\"$input.params('stream-name')\"\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"delete": {
  "parameters": [
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "schema": {
        "type": "string"
      }
    }
  ]
},
"responses": {
  "200": {
    "description": "200 response",
    "headers": {
      "Content-Type": {
        "schema": {
          "type": "string"
        }
      }
    }
  },
  "content": {
    "application/json": {
      "schema": {
        "$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",

```



```
}
```

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\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\":\n    \"$input.params('stream-name')\"\n    }\n    },\n    \"passthroughBehavior\": \"when_no_match\",\n    \"httpMethod\": \"POST\"\n  }\n},\n\"delete\": {\n  \"consumes\": [\n    \"application/json\"\n  ],\n  \"produces\": [\n    \"application/json\"\n  ],\n  \"parameters\": [\n    {\n      \"name\": \"stream-name\",\n      \"in\": \"path\",\n      \"required\": true,\n      \"type\": \"string\"\n    }\n  ],\n  \"responses\": {\n    \"200\": {\n      \"description\": \"200 response\",\n      \"schema\": {\n        \"$ref\": \"#/definitions/Empty\"\n      },\n      \"headers\": {\n        \"Content-Type\": {\n          \"type\": \"string\"\n        }\n      }\n    },\n    \"400\": {\n      \"description\": \"400 response\",\n      \"headers\": {\n        \"Content-Type\": {\n          \"type\": \"string\"\n        }\n      }\n    },\n    \"500\": {\n      \"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    \"StreamName\": \"$input.params('stream-
name')\"\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  \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\n                : \"${elem.data}\",\n            \"PartitionKey\" : \"${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      \"ShardId\": \"${input.params('shard-id')}\",\n      \"ShardIteratorType\": \"TRIM_HORIZON\",\n      \"StreamName\": \"${input.params('stream-name')}\",\n      \"passthroughBehavior\": \"when_no_match\",\n      \"httpMethod\": \"POST\"\n    }\n  }\n},\n\"definitions\": {\n  \"Empty\": {\n    \"type\": \"object\"\n  },\n  \"PutRecordsMethodRequestPayload\": {\n    \"type\": \"object\",\n    \"properties\": {\n      \"records\": {\n        \"type\": \"array\",\n        \"items\": {\n          \"type\": \"object\",\n          \"properties\": {\n            \"data\": {\n              \"type\": \"string\"\n            },\n            \"partition-key\": {\n              \"type\": \"string\"\n            }\n          }\n        }\n      }\n    }\n  }\n}\n}

```

Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI

Gunakan tutorial berikut untuk membuat PetStore API yang mendukung GET /pets dan GET /pets/{petId} metode. Metode terintegrasi dengan titik akhir HTTP. Anda dapat mengikuti tutorial ini menggunakan AWS SDK for JavaScript, SDK for Python (Boto3), atau file. AWS CLI Anda menggunakan fungsi atau perintah berikut untuk menyiapkan API Anda:

JavaScript v3

- [CreateRestApiCommand](#)
- [CreateResourceCommand](#)
- [PutMethodCommand](#)
- [PutMethodResponseCommand](#)
- [PutIntegrationCommand](#)
- [PutIntegrationResponseCommand](#)
- [CreateDeploymentCommand](#)

Python

- [create_rest_api](#)
- [create_resource](#)
- [put_metode](#)
- [put_method_response](#)
- [put_integrasi](#)
- [put_integration_response](#)
- [create_deployment](#)

AWS CLI

- [create-rest-api](#)
- [buat-sumber daya](#)
- [metode put-](#)
- [put-method-response](#)
- [put-integrasi](#)
- [put-integration-response](#)
- [membuat-penyebaran](#)

Untuk informasi selengkapnya tentang AWS SDK untuk JavaScript v3, lihat Untuk [apa AWS SDK itu? JavaScript](#) . Untuk informasi selengkapnya tentang SDK for Python (Boto3), lihat. [AWS SDK untuk Python \(Boto3\)](#) Untuk informasi lebih lanjut tentang AWS CLI, lihat [Apa itu AWS CLI?](#) .

Siapkan API yang dioptimalkan tepi PetStore

Dalam tutorial ini, contoh perintah menggunakan nilai placeholder untuk nilai IDs seperti ID API dan ID sumber daya. Saat Anda menyelesaikan tutorial, ganti nilai-nilai ini dengan milik Anda sendiri.

Untuk menyiapkan API yang dioptimalkan tepi PetStore menggunakan AWS SDKs

1. Gunakan contoh berikut untuk membuat RestApi entitas:

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)
  }
})();
```

Panggilan yang berhasil mengembalikan ID API Anda dan ID sumber daya root API Anda dalam output seperti berikut:

```
{
  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 boto3
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","createdDate","version","binaryMediaTypes","apiKeySource"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Panggilan yang berhasil mengembalikan ID API Anda dan ID sumber daya root API Anda dalam output seperti berikut:

```
{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
'Demo API created using the AWS SDK for Python', 'createdDate':
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
```

Berikut ini adalah output dari perintah ini:

```
{
  "id": "abcd1234",
  "name": "Simple PetStore (AWS CLI)",
  "createdDate": "2022-12-15T08:07:04-08:00",
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "EDGE"
    ]
  },
  "disableExecuteApiEndpoint": false,
  "rootResourceId": "efg567"
}
```

API yang Anda buat memiliki ID API abcd1234 dan ID sumber daya root dari efg567. Anda menggunakan nilai-nilai ini dalam pengaturan API Anda.

2. Selanjutnya, Anda menambahkan sumber daya anak di bawah root, Anda menentukan RootResourceId sebagai nilai parentId properti. Gunakan contoh berikut untuk membuat /pets sumber daya untuk API Anda:

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);
  }
})();
```

Panggilan yang berhasil mengembalikan informasi tentang sumber daya Anda dalam output seperti berikut:

```
{
  "path": "/pets",
  "pathPart": "pets",
  "id": "aaa111",
  "parentId": "efg567'"
}
```

Python

```
import boto3
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)
```

Panggilan yang berhasil mengembalikan informasi tentang sumber daya Anda dalam output seperti berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{  
  "id": "aaa111",  
  "parentId": "efg567",  
  "pathPart": "pets",  
  "path": "/pets"  
}
```

Sumber /pets daya yang Anda buat memiliki ID sumber dayaaaa111. Anda menggunakan nilai ini dalam pengaturan API Anda.

3. Selanjutnya, Anda menambahkan sumber daya anak di bawah sumber /pets daya. Sumber daya ini, /{petId} memiliki parameter jalur {petId} untuk .To membuat bagian jalur sebagai parameter jalur, lampirkan dalam sepasang tanda kurung kurawal,. { } Gunakan contoh berikut untuk membuat /pets/{petId} sumber daya untuk API Anda:

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)  
  }  
})();
```

Panggilan yang berhasil mengembalikan informasi tentang sumber daya Anda dalam output seperti berikut:

```
{
  "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)
```

Panggilan yang berhasil mengembalikan informasi tentang sumber daya Anda dalam output seperti berikut:

```
{'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}'
```

Berikut ini adalah output dari perintah ini:

```
{  
  "id": "bbb222",  
  "parentId": "aaa111",  
  "path": "/pets/{petId}",  
  "pathPart": "{petId}"  
}
```

Sumber `/pets/{petId}` daya yang Anda buat memiliki ID sumber `dayabbb222`. Anda menggunakan nilai ini dalam pengaturan API Anda.

4. Dalam dua langkah berikut, Anda menambahkan metode HTTP ke sumber daya Anda. Dalam tutorial ini, Anda mengatur metode untuk memiliki akses terbuka dengan mengatur `authorization-type` to set `noAuthorization`. Untuk mengizinkan hanya pengguna yang diautentikasi untuk memanggil metode, Anda dapat menggunakan peran dan kebijakan IAM, otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus), atau kumpulan pengguna Amazon Cognito. Untuk informasi selengkapnya, lihat [the section called “Kontrol akses”](#).

Gunakan contoh berikut untuk menambahkan metode GET HTTP pada `/pets` sumber daya:

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)
}
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE"
}
```

Python

```
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)
```

Panggilan yang berhasil mengembalikan output berikut:


```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{  
  "httpMethod": "GET",  
  "authorizationType": "NONE",  
  "apiKeyRequired": false  
}
```

- Gunakan contoh berikut untuk menambahkan metode GET HTTP pada `/pets/{petId}` sumber daya dan untuk mengatur `requestParameters` properti untuk meneruskan `petId` nilai yang disediakan klien ke backend:

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)
    }
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "apiKeyRequired": false,
  "requestParameters": {
    "method.request.path.petId": true
  }
}
```

- Gunakan contoh berikut untuk menambahkan respons metode 200 OK untuk GET /pets metode ini:

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)
    }
}
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{  
  "statusCode": "200"  
}
```

- Gunakan contoh berikut untuk menambahkan respons metode 200 OK untuk GET `/pets/{petId}` metode ini:

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)  
  }  
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{  
  "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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{
  "statusCode": "200"
}
```

- Gunakan contoh berikut untuk mengkonfigurasi integrasi untuk GET /pets metode dengan titik akhir HTTP. Endpoint HTTP adalah `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)
}
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{
  "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. Gunakan contoh berikut untuk mengkonfigurasi integrasi untuk GET /pets/{petId} metode dengan titik akhir HTTP. Endpoint HTTP adalah `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`. Pada langkah ini, Anda memetakan parameter jalur `petId` ke parameter jalur titik akhir integrasi. `id`

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)
  }
})();

```

Panggilan yang berhasil mengembalikan output berikut:

```
{
```

```

    "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)

```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{
  "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. Gunakan contoh berikut untuk menambahkan respons integrasi untuk GET /pets integrasi:

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)
}
})();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{
  "statusCode": "200",
  "selectionPattern": ""
}
```

11. Gunakan contoh berikut untuk menambahkan respons integrasi untuk GET /pets/{petId} integrasi:

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)
    }
    })();
```

Panggilan yang berhasil mengembalikan output berikut:

```
{
  "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='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)
```

Panggilan yang berhasil mengembalikan output berikut:

```
{'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
```

Berikut ini adalah output dari perintah ini:

```
{  
  "statusCode": "200",  
  "selectionPattern": ""  
}
```

Setelah Anda membuat respons integrasi, API Anda dapat menanyakan hewan peliharaan yang tersedia di PetStore situs web dan untuk melihat hewan peliharaan individu dari pengenal tertentu. Sebelum API Anda dapat dipanggil oleh pelanggan Anda, Anda harus menerapkannya. Sebaiknya sebelum menerapkan API, Anda mengujinya.

- Gunakan contoh berikut untuk menguji GET /pets metode:

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. Gunakan contoh berikut untuk menguji GET /pets/{petId} metode dengan petId 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 boto3
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'
```

Setelah berhasil menguji API, Anda dapat menerapkannya ke panggung.

- Gunakan contoh berikut untuk menerapkan API Anda ke tahap bernama `test`. Saat Anda menerapkan API ke sebuah panggung, pemanggil API dapat memanggil API Anda.

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 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'
```

Berikut ini adalah output dari perintah ini:

```
{  
  "id": "ab1c1d",  
  "description": "First deployment",  
  "createdDate": "2022-12-15T08:44:13-08:00"  
}
```

API Anda sekarang dapat dipanggil oleh pelanggan. Anda dapat menguji API ini dengan memasukkan `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets` URL di browser, dan menggantinya `abcd1234` dengan pengenalan API Anda.

Untuk contoh selengkapnya tentang cara membuat atau memperbarui API menggunakan AWS SDKs atau menggunakan API AWS CLI, lihat [Tindakan untuk API Gateway menggunakan AWS SDKs](#).

Otomatisasikan penyiapan API Anda

Alih-alih membuat API step-by-step, Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan OpenAPI, CloudFormation, atau Terraform untuk membuat API Anda.

Definisi OpenAPI 3.0

Anda dapat mengimpor definisi OpenAPI ke API Gateway. Untuk informasi selengkapnya, lihat [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 Template

Untuk menerapkan CloudFormation template Anda, lihat [Membuat tumpukan di AWS CloudFormation konsol](#).

```

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'

Konfigurasi Terraform

[Untuk informasi lebih lanjut tentang Terraform, lihat 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
  }

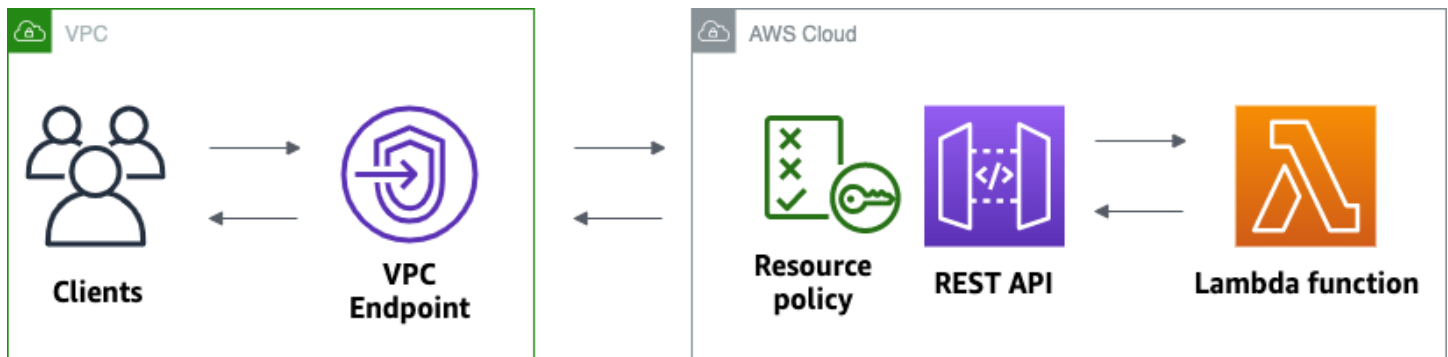
  resource "aws_api_gateway_deployment" "Deployment" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    depends_on =
      [aws_api_gateway_integration.petsIntegration,aws_api_gateway_integration.petIdIntegration ]
  }
  resource "aws_api_gateway_stage" "Stage" {
    stage_name      = "Prod"
    rest_api_id     = aws_api_gateway_rest_api.Api.id
    deployment_id   = aws_api_gateway_deployment.Deployment.id
  }

```

Tutorial: Membuat REST API pribadi

Dalam tutorial ini, Anda membuat REST API pribadi. Klien dapat mengakses API hanya dari dalam VPC Amazon Anda. API diisolasi dari internet publik, yang merupakan persyaratan keamanan umum.

Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya. Pertama, Anda menggunakan CloudFormation template untuk membuat VPC Amazon, titik akhir VPC, AWS Lambda fungsi, dan meluncurkan instans Amazon EC2 yang akan Anda gunakan untuk menguji API Anda. Selanjutnya, Anda menggunakan Konsol Manajemen AWS untuk membuat API pribadi dan melampirkan kebijakan sumber daya yang memungkinkan akses hanya dari titik akhir VPC Anda. Terakhir, Anda menguji API Anda.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Dalam tutorial ini, Anda menggunakan Konsol Manajemen AWS. Untuk CloudFormation template yang membuat API ini dan semua resource terkait, lihat [template.yaml](#).

Topik

- [Langkah 1: Buat dependensi](#)
- [Langkah 2: Buat API pribadi](#)
- [Langkah 3: Buat metode dan integrasi](#)
- [Langkah 4: Lampirkan kebijakan sumber daya](#)
- [Langkah 5: Menerapkan API Anda](#)
- [Langkah 6: Verifikasi bahwa API Anda tidak dapat diakses publik](#)
- [Langkah 7: Hubungkan ke instans di VPC Anda dan panggil API Anda](#)
- [Langkah 8: Membersihkan](#)
- [Langkah selanjutnya: Otomatisasi dengan CloudFormation](#)

Langkah 1: Buat dependensi

Unduh dan unzip [CloudFormation template ini](#). Anda menggunakan template untuk membuat semua dependensi untuk API pribadi Anda, termasuk VPC Amazon, titik akhir VPC, dan fungsi Lambda yang berfungsi sebagai backend API Anda. Anda membuat API pribadi nanti.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).

3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, **private-api-tutorial** masukkan lalu pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation menyediakan dependensi untuk API Anda, yang dapat memakan waktu beberapa menit. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, pilih Output. Perhatikan ID titik akhir VPC Anda. Anda membutuhkannya untuk langkah-langkah selanjutnya dalam tutorial ini.

Langkah 2: Buat API pribadi

Anda membuat API pribadi untuk mengizinkan hanya klien dalam VPC Anda untuk mengaksesnya.

Untuk membuat API pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Create API, lalu untuk REST API, pilih Build.
3. Untuk nama API, masukkan **private-api-tutorial**.
4. Untuk jenis endpoint API, pilih Private.
5. Untuk titik akhir VPC IDs, masukkan ID titik akhir VPC dari Output tumpukan Anda. CloudFormation
6. Untuk jenis alamat IP, pilih Dualstack.
7. Pilih Buat API.

Langkah 3: Buat metode dan integrasi

Anda membuat GET metode dan integrasi Lambda untuk menangani GET permintaan ke API Anda. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda yang Anda buat di Langkah 1, lalu mengembalikan respons ke klien.

Untuk membuat metode dan integrasi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih metode Buat.
4. Untuk jenis Metode pilih GET.
5. Untuk jenis Integrasi, pilih fungsi Lambda.
6. Aktifkan integrasi proxy Lambda. Dengan integrasi proxy Lambda, API Gateway mengirimkan peristiwa ke Lambda dengan struktur yang ditentukan, dan mengubah respons dari fungsi Lambda Anda menjadi respons HTTP.
7. Untuk fungsi Lambda, pilih fungsi yang Anda buat dengan CloudFormation template di Langkah 1. Nama fungsi dimulai dengan **private-api-tutorial**.
8. Pilih metode Buat.

Langkah 4: Lampirkan kebijakan sumber daya

Anda melampirkan [kebijakan sumber daya](#) ke API Anda yang memungkinkan klien menjalankan API hanya melalui titik akhir VPC Anda. Untuk lebih membatasi akses ke API Anda, Anda juga dapat mengonfigurasi kebijakan titik akhir [VPC untuk titik akhir](#) VPC Anda, tetapi itu tidak diperlukan untuk tutorial ini.

Untuk melampirkan kebijakan sumber daya

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Kebijakan sumber daya, lalu pilih Buat kebijakan.
4. Masukkan kebijakan berikut. Ganti *vpceID* dengan ID titik akhir VPC Anda dari Output tumpukan Anda. CloudFormation

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. Pilih Simpan perubahan.

Langkah 5: Menerapkan API Anda

Selanjutnya, Anda menerapkan API Anda untuk membuatnya tersedia bagi klien di Amazon VPC Anda.

Untuk menerapkan API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Deploy API.
4. Untuk Stage, pilih New stage.
5. Untuk nama Panggung, masukkan **test**.
6. (Opsional) Untuk Deskripsi, masukkan deskripsi.
7. Pilih Deploy.

Sekarang Anda siap untuk menguji API Anda.

Langkah 6: Verifikasi bahwa API Anda tidak dapat diakses publik

Gunakan `curl` untuk memverifikasi bahwa Anda tidak dapat menjalankan API dari luar VPC Amazon Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap pengujian.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda. URL terlihat seperti `https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. Titik akhir VPC yang Anda buat di Langkah 1 memiliki DNS pribadi yang diaktifkan, sehingga Anda dapat menggunakan URL yang disediakan untuk menjalankan API Anda.
5. Gunakan `curl` untuk mencoba menjalankan API Anda dari luar VPC Anda.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Curl menunjukkan bahwa titik akhir API Anda tidak dapat diselesaikan. Jika Anda mendapatkan respons yang berbeda, kembali ke Langkah 2, dan pastikan Anda memilih Private untuk jenis endpoint API Anda.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/  
test
```

Selanjutnya, Anda terhubung ke instans Amazon EC2 di VPC Anda untuk menjalankan API Anda.

Langkah 7: Hubungkan ke instans di VPC Anda dan panggil API Anda

Selanjutnya, Anda menguji API Anda dari dalam VPC Amazon Anda. Untuk mengakses API pribadi, sambungkan ke instans Amazon EC2 di VPC, lalu gunakan `curl` untuk menjalankan API. Anda menggunakan Systems Manager Session Manager untuk menyambung ke instans Anda di browser.

Untuk menguji API Anda

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Pilih Instans.

3. Pilih contoh bernama `private-api-tutorial` yang Anda buat dengan CloudFormation template di Langkah 1.
4. Pilih Connect dan kemudian pilih Session Manager.
5. Pilih Connect untuk meluncurkan sesi berbasis browser ke instans Anda.
6. Di sesi Session Manager Anda, gunakan `curl` untuk menjalankan API Anda. Anda dapat menjalankan API karena Anda menggunakan instance di VPC Amazon Anda.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Verifikasi bahwa Anda mendapatkan `responHello from Lambda!`.



Anda berhasil membuat API yang hanya dapat diakses dari dalam VPC Amazon Anda dan kemudian memverifikasi bahwa itu berfungsi.

Langkah 8: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus REST API dan CloudFormation tumpukan Anda.

Untuk menghapus REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada API salaman, pilih API. Pilih tindakan API, pilih Hapus API, lalu konfirmasi pilihan Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Langkah selanjutnya: Otomatisasi dengan CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua AWS sumber daya yang terlibat dalam tutorial ini. Untuk contoh CloudFormation template lengkap, lihat [template.yaml](#).

Tutorial API HTTP Amazon API Gateway

Tutorial berikut menyediakan latihan langsung untuk membantu Anda mempelajari tentang API Gateway HTTP. APIs

Topik

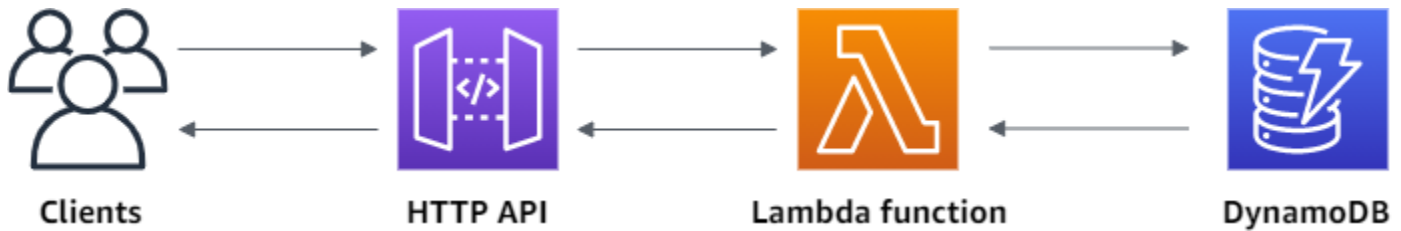
- [Tutorial: Buat API HTTP CRUD dengan Lambda dan DynamoDB](#)
- [Tutorial: Buat API HTTP dengan integrasi pribadi ke layanan Amazon ECS](#)

Tutorial: Buat API HTTP CRUD dengan Lambda dan DynamoDB

Dalam tutorial ini, Anda membuat API tanpa server yang membuat, membaca, memperbarui, dan menghapus item dari tabel DynamoDB. DynamoDB adalah layanan basis data NoSQL terkelola penuh yang memberikan performa yang cepat dan dapat diprediksi dengan skalabilitas sempurna. Tutorial ini membutuhkan waktu sekitar 30 menit untuk menyelesaikannya, dan Anda dapat melakukannya dalam [Tingkat AWS Gratis](#).

Pertama, Anda membuat tabel [DynamoDB](#) menggunakan konsol DynamoDB. Kemudian Anda membuat fungsi [Lambda](#) menggunakan konsol. AWS Lambda Selanjutnya, Anda membuat API HTTP menggunakan konsol API Gateway. Terakhir, Anda menguji API Anda.

Saat Anda memanggil API HTTP, API Gateway merutekan permintaan ke fungsi Lambda Anda. Fungsi Lambda berinteraksi dengan DynamoDB, dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Dalam tutorial ini, Anda menggunakan Konsol Manajemen AWS. Untuk AWS SAM template yang membuat API ini dan semua sumber daya terkait, lihat [template.yaml](#).

Topik

- [Langkah 1: Buat tabel DynamoDB](#)
- [Langkah 2: Buat fungsi Lambda](#)
- [Langkah 3: Buat API HTTP](#)
- [Langkah 4: Buat rute](#)
- [Langkah 5: Buat integrasi](#)
- [Langkah 6: Lampirkan integrasi Anda ke rute](#)
- [Langkah 7: Uji API Anda](#)
- [Langkah 8: Membersihkan](#)
- [Langkah selanjutnya: Otomatiskan dengan AWS SAM atau CloudFormation](#)

Langkah 1: Buat tabel DynamoDB

Anda menggunakan tabel [DynamoDB](#) untuk menyimpan data untuk API Anda.

Setiap item memiliki ID unik, yang kita gunakan sebagai [kunci partisi](#) untuk tabel.

Untuk membuat tabel DynamoDB

1. Buka konsol DynamoDB di <https://console.aws.amazon.com/dynamodb/>
2. Pilih Buat tabel.
3. Untuk Nama tabel, masukkan **http-crud-tutorial-items**.
4. Untuk kunci Partisi, masukkan **id**.
5. Pilih Buat tabel.

Langkah 2: Buat fungsi Lambda

Anda membuat fungsi [Lambda](#) untuk backend API Anda. Fungsi Lambda ini membuat, membaca, memperbarui, dan menghapus item dari DynamoDB. Fungsi ini menggunakan [peristiwa dari API Gateway](#) untuk menentukan cara berinteraksi dengan DynamoDB. Untuk kesederhanaan tutorial ini menggunakan fungsi Lambda tunggal. Sebagai praktik terbaik, Anda harus membuat fungsi terpisah untuk setiap rute. Untuk informasi lebih lanjut, lihat [Monolit Lambda](#).

Untuk membuat fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pilih Buat fungsi.
3. Untuk Nama fungsi, masukkan **http-crud-tutorial-function**.
4. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
5. Di bawah Izin pilih Ubah peran eksekusi default.
6. Pilih Buat peran baru dari templat AWS kebijakan.
7. Untuk Nama peran, masukkan **http-crud-tutorial-role**.
8. Untuk templat Kebijakan, pilih **Simple microservice permissions**. Kebijakan ini memberikan izin fungsi Lambda untuk berinteraksi dengan DynamoDB.

Note

Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

9. Pilih Buat fungsi.
10. Buka fungsi Lambda di editor kode konsol, dan ganti isinya dengan kode berikut. Pilih Deploy untuk memperbarui fungsi Anda.

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
```

Langkah 3: Buat API HTTP

HTTP API menyediakan endpoint HTTP untuk fungsi Lambda Anda. Pada langkah ini, Anda membuat API kosong. Pada langkah-langkah berikut, Anda mengonfigurasi rute dan integrasi untuk menghubungkan API dan fungsi Lambda Anda.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API, lalu untuk HTTP API, pilih Build.
3. Untuk nama API, masukkan **http-crud-tutorial-api**.
4. Untuk jenis alamat IP, pilih IPv4.
5. Pilih Berikutnya.
6. Untuk Konfigurasi rute, pilih Berikutnya untuk melewati pembuatan rute. Anda membuat rute nanti.
7. Tinjau tahap yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
8. Pilih Buat.

Langkah 4: Buat rute

Rute adalah cara untuk mengirim permintaan API yang masuk ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya, misalnya, GET /items. Untuk contoh API ini, kita membuat empat rute:

- GET /items/{id}
- GET /items
- PUT /items
- DELETE /items/{id}

Untuk membuat rute

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Rute.
4. Pilih Buat.
5. Untuk Metode, pilih **GET**.
6. Untuk jalan, masuk **/items/{id}**. {id} Di akhir jalur adalah parameter jalur yang diambil API Gateway dari jalur permintaan saat klien membuat permintaan.
7. Pilih Buat.
8. Ulangi langkah 4-7 untuk **GET /items**, **DELETE /items/{id}**, dan **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

Langkah 5: Buat integrasi

Anda membuat integrasi untuk menghubungkan rute ke sumber daya backend. Untuk API contoh ini, Anda membuat satu integrasi Lambda yang Anda gunakan untuk semua rute.

Untuk membuat integrasi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih Kelola integrasi dan kemudian pilih Buat.
5. Lewati Lampirkan integrasi ini ke rute. Anda menyelesaikannya di langkah selanjutnya.
6. Untuk jenis Integrasi, pilih fungsi Lambda.
7. Untuk fungsi Lambda, masukkan. **http-crud-tutorial-function**
8. Pilih Buat.

Langkah 6: Lampirkan integrasi Anda ke rute

Untuk API contoh ini, Anda menggunakan integrasi Lambda yang sama untuk semua rute. Setelah Anda melampirkan integrasi ke semua rute API, fungsi Lambda Anda akan dipanggil saat klien memanggil salah satu rute Anda.

Untuk melampirkan integrasi ke rute

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih rute.
5. Di bawah Pilih integrasi yang ada, pilih **http-crud-tutorial-function**.
6. Pilih Lampirkan integrasi.
7. Ulangi langkah 4-6 untuk semua rute.

Semua rute menunjukkan bahwa AWS Lambda integrasi terlampir.

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

Sekarang Anda memiliki API HTTP dengan rute dan integrasi, Anda dapat menguji API Anda.

Langkah 7: Uji API Anda

Untuk memastikan bahwa API Anda berfungsi, Anda menggunakan [curl](#).

Untuk mendapatkan URL untuk menjalankan API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Perhatikan URL pemanggilan API Anda. Itu muncul di bawah Memanggil URL pada halaman Detail.

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)

Find resources

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. Salin URL pemanggilan API Anda.

URL lengkapnya terlihat seperti `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Untuk membuat atau memperbarui item

- Gunakan perintah berikut untuk membuat atau memperbarui item. Perintah tersebut mencakup badan permintaan dengan ID, harga, dan nama item.

```
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
```

Untuk mendapatkan semua item

- Gunakan perintah berikut untuk daftar semua item.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Untuk mendapatkan item

- Gunakan perintah berikut untuk mendapatkan item dengan ID-nya.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Untuk menghapus item

1. Gunakan perintah berikut untuk menghapus item.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Dapatkan semua item untuk memverifikasi bahwa item telah dihapus.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Langkah 8: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari latihan memulai ini. Langkah-langkah berikut menghapus API HTTP, fungsi Lambda, dan sumber daya terkait.

Untuk menghapus tabel DynamoDB

1. Buka konsol DynamoDB di <https://console.aws.amazon.com/dynamodb/>
2. Pilih meja Anda.
3. Pilih Hapus tabel.
4. Konfirmasikan pilihan Anda, dan pilih Hapus.

Untuk menghapus API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada APIshalaman, pilih API. Pilih Tindakan, lalu pilih Hapus.
3. Pilih Hapus.

Untuk menghapus fungsi Lambda

1. [Masuk ke konsol Lambda di https://console.aws.amazon.com /lambda](https://console.aws.amazon.com /lambda).
2. Pada halaman Fungsi, pilih fungsi. Pilih Tindakan, lalu pilih Hapus.
3. Pilih Hapus.

Untuk menghapus grup log fungsi Lambda

1. Di CloudWatch konsol Amazon, buka [halaman Grup log](#).
2. Pada halaman Grup log, pilih grup log fungsi (/aws/lambda/http-crud-tutorial-function). Pilih Tindakan, lalu pilih Hapus grup log.
3. Pilih Hapus.

Untuk menghapus peran eksekusi fungsi Lambda

1. Di AWS Identity and Access Management konsol, buka [halaman Peran](#).
2. Pilih peran fungsi, misalnya, http-crud-tutorial-role.
3. Pilih Hapus peran.
4. Pilih Ya, Hapus.

Langkah selanjutnya: Otomatiskan dengan AWS SAM atau CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan CloudFormation atau AWS SAM. Untuk contoh AWS SAM template untuk tutorial ini, lihat [template.yaml](#).

Misalnya CloudFormation template, lihat [contoh CloudFormation template](#).

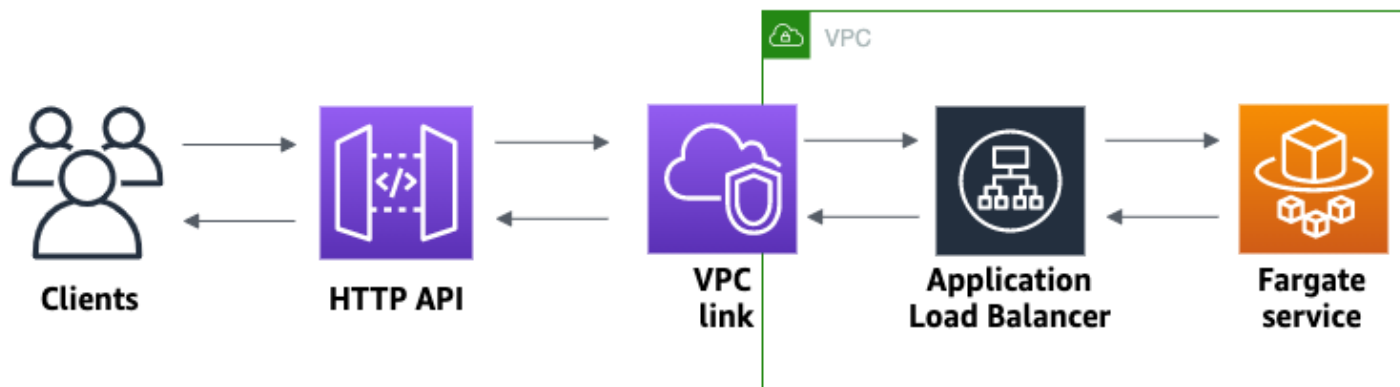
Tutorial: Buat API HTTP dengan integrasi pribadi ke layanan Amazon ECS

Dalam tutorial ini, Anda membuat API tanpa server yang terhubung ke layanan Amazon ECS yang berjalan di VPC Amazon. Klien di luar VPC Amazon Anda dapat menggunakan API untuk mengakses layanan Amazon ECS Anda.

Tutorial ini membutuhkan waktu sekitar satu jam untuk menyelesaikannya. Pertama, Anda menggunakan CloudFormation template untuk membuat Amazon VPC dan layanan Amazon ECS.

Kemudian Anda menggunakan konsol API Gateway untuk membuat tautan VPC. Tautan VPC memungkinkan API Gateway untuk mengakses layanan Amazon ECS yang berjalan di VPC Amazon Anda. Selanjutnya, Anda membuat API HTTP yang menggunakan tautan VPC untuk terhubung ke layanan Amazon ECS Anda. Terakhir, Anda menguji API Anda.

Saat Anda menjalankan API HTTP, API Gateway merutekan permintaan ke layanan Amazon ECS Anda melalui tautan VPC Anda, lalu mengembalikan respons dari layanan.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Dalam tutorial ini, Anda menggunakan Konsol Manajemen AWS. Untuk CloudFormation template yang membuat API ini dan semua resource terkait, lihat [template.yaml](#).

Topik

- [Langkah 1: Buat layanan Amazon ECS](#)
- [Langkah 2: Buat tautan VPC](#)
- [Langkah 3: Buat API HTTP](#)
- [Langkah 4: Buat rute](#)
- [Langkah 5: Buat integrasi](#)
- [Langkah 6: Uji API Anda](#)
- [Langkah 7: Bersihkan](#)
- [Langkah selanjutnya: Otomatiskan dengan CloudFormation](#)

Langkah 1: Buat layanan Amazon ECS

Amazon ECS adalah layanan manajemen kontainer yang memudahkan untuk menjalankan, menghentikan, dan mengelola kontainer Docker di cluster. Dalam tutorial ini, Anda menjalankan cluster Anda pada infrastruktur tanpa server yang dikelola oleh Amazon ECS.

Unduh dan unzip [CloudFormation template ini](#), yang membuat semua dependensi untuk layanan, termasuk VPC Amazon. Anda menggunakan template untuk membuat layanan Amazon ECS yang menggunakan Application Load Balancer.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **http-api-private-integrations-tutorial** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation menyediakan layanan ECS, yang dapat memakan waktu beberapa menit. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat tautan VPC

Tautan VPC memungkinkan API Gateway untuk mengakses sumber daya pribadi di VPC Amazon. Anda menggunakan tautan VPC untuk memungkinkan klien mengakses layanan Amazon ECS Anda melalui HTTP API Anda.

Untuk membuat tautan VPC

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pada panel navigasi utama, pilih tautan VPC lalu pilih Buat.

Anda mungkin perlu memilih ikon menu untuk membuka panel navigasi utama.

3. Untuk Pilih versi tautan VPC, pilih tautan VPC untuk HTTP. APIs
4. Untuk Nama, masukkan **private-integrations-tutorial**.
5. Untuk VPC, pilih VPC yang Anda buat di langkah 1. Nama harus dimulai dengan PrivateIntegrationsStack.
6. Untuk Subnet, pilih dua subnet pribadi di VPC Anda. Nama mereka diakhiri dengan PrivateSubnet.
7. Untuk grup Keamanan, pilih ID Grup yang dimulai dengan private-integrations-tutorial dan memiliki deskripsi PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup.
8. Pilih Buat.

Setelah Anda membuat tautan VPC, API Gateway menyediakan Antarmuka Jaringan Elastis untuk mengakses VPC Anda. Prosesnya bisa memakan waktu beberapa menit. Sementara itu, Anda dapat membuat API Anda.

Langkah 3: Buat API HTTP

HTTP API menyediakan endpoint HTTP untuk layanan Amazon ECS Anda. Pada langkah ini, Anda membuat API kosong. Pada Langkah 4 dan 5, Anda mengonfigurasi rute dan integrasi untuk menghubungkan API dan layanan Amazon ECS Anda.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API, lalu untuk HTTP API, pilih Build.
3. Untuk nama API, masukkan **http-private-integrations-tutorial**.
4. Untuk jenis alamat IP, pilih IPv4.
5. Pilih Berikutnya.
6. Untuk Mengonfigurasi rute, pilih Berikutnya untuk melewati pembuatan rute. Anda membuat rute nanti.

7. Tinjau tahap yang dibuat API Gateway untuk Anda. API Gateway membuat `$default` panggung dengan penerapan otomatis diaktifkan, yang merupakan pilihan terbaik untuk tutorial ini. Pilih Berikutnya.
8. Pilih Buat.

Langkah 4: Buat rute

Rute adalah cara untuk mengirim permintaan API yang masuk ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya, misalnya, `GET /items`. Untuk contoh API ini, kita membuat satu rute.

Untuk membuat rute

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Rute.
4. Pilih Buat.
5. Untuk Metode, pilih **ANY**.
6. Untuk jalan, masuk **`/{{proxy+}}`**. `{{proxy+}}` Di ujung jalan adalah variabel jalur serakah. API Gateway mengirimkan semua permintaan ke API Anda ke rute ini.
7. Pilih Buat.

Langkah 5: Buat integrasi

Anda membuat integrasi untuk menghubungkan rute ke sumber daya backend.

Untuk membuat integrasi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih Kelola integrasi dan kemudian pilih Buat.
5. Untuk Lampirkan integrasi ini ke rute, pilih rute **ANY/{{proxy+}}** yang Anda buat sebelumnya.
6. Untuk jenis Integrasi, pilih Sumber daya pribadi.
7. Untuk detail Integrasi, pilih Pilih secara manual.

8. Untuk layanan Target, pilih ALB/NLB.
9. Untuk Load balancer, pilih load balancer yang Anda buat dengan CloudFormation template di Langkah 1. Namanya harus dimulai dengan HTTP-Priva.
10. Untuk pendengar, pilih **HTTP 80**.
11. Untuk tautan VPC, pilih tautan VPC yang Anda buat di Langkah 2. Seharusnya namanya `private-integrations-tutorial`.
12. Pilih Buat.

Untuk memverifikasi bahwa rute dan integrasi Anda telah diatur dengan benar, pilih Lampirkan integrasi ke rute. Konsol menunjukkan bahwa Anda memiliki ANY `/[proxy+]` rute dengan integrasi ke Load Balancer VPC.

Integrations

Stage: - ▼

Deploy

Attach integrations to routes

Manage integrations

Routes for private-integrations-tutorial

Search

▼ `/[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

Sekarang Anda siap untuk menguji API Anda.

Langkah 6: Uji API Anda

Selanjutnya, Anda menguji API Anda untuk memastikan bahwa itu berfungsi. Untuk mempermudah, gunakan browser web untuk menjalankan API Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Perhatikan URL pemanggilan API Anda.

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. Di browser web, buka URL pemanggilan API Anda.

URL lengkap akan terlihat seperti `https://abcdef123.execute-api.us-east-2.amazonaws.com`.

Browser Anda mengirimkan GET permintaan ke API.

5. Verifikasi bahwa respons API Anda adalah pesan selamat datang yang memberi tahu Anda bahwa aplikasi Anda berjalan di Amazon ECS.

Jika Anda melihat pesan selamat datang, Anda berhasil membuat layanan Amazon ECS yang berjalan di VPC Amazon, dan Anda menggunakan API API Gateway HTTP dengan tautan VPC untuk mengakses layanan Amazon ECS.

Langkah 7: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus tautan VPC, CloudFormation tumpukan, dan API HTTP Anda.

Untuk menghapus API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada APIshalaman, pilih API. Pilih Tindakan, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus tautan VPC

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih tautan VPC.
3. Pilih tautan VPC Anda, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Langkah selanjutnya: Otomatiskan dengan CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua AWS sumber daya yang terlibat dalam tutorial ini. Untuk contoh CloudFormation template lengkap, lihat [template.yaml](#).

Tutorial API Amazon WebSocket API Gateway

Tutorial berikut menyediakan latihan langsung untuk membantu Anda mempelajari tentang API Gateway. WebSocket APIs

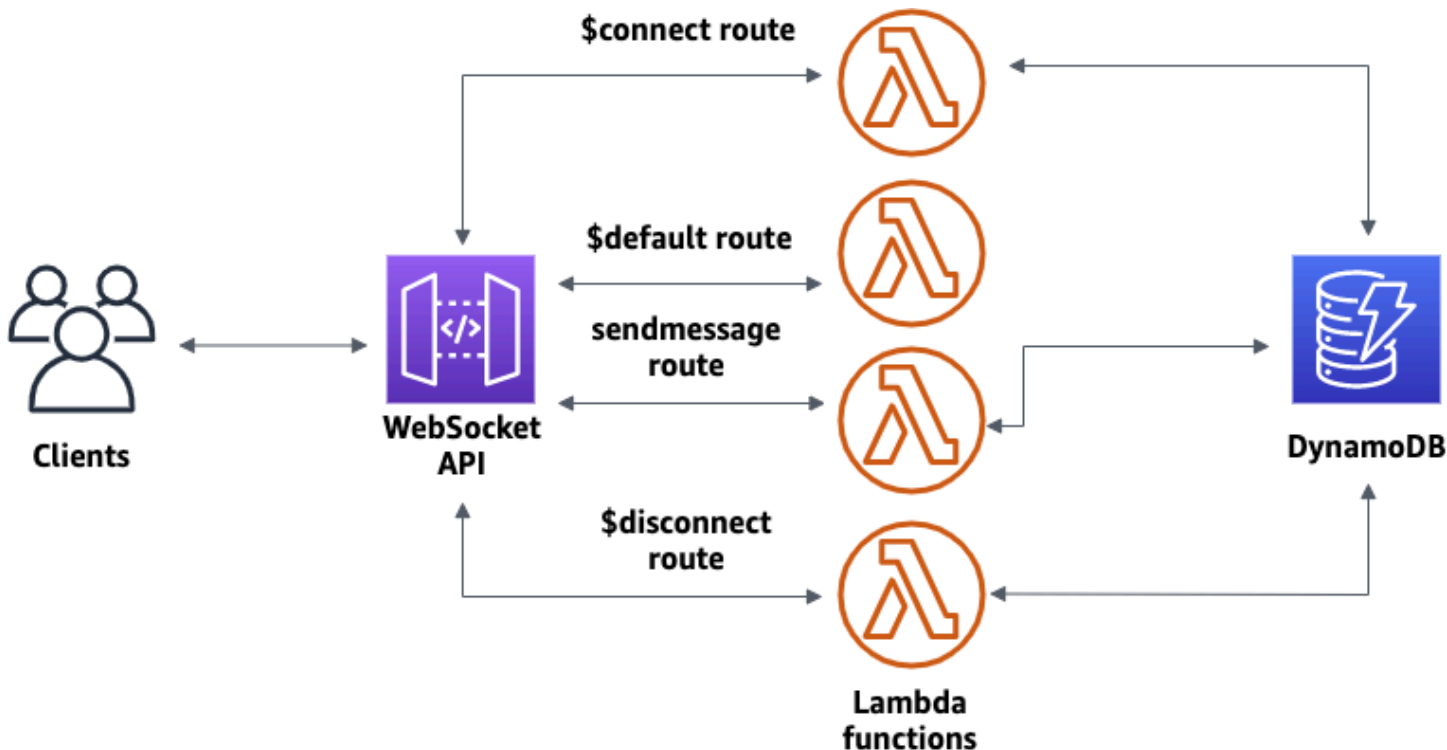
Topik

- [Tutorial: Buat aplikasi WebSocket obrolan dengan WebSocket API, Lambda, dan DynamoDB](#)
- [Tutorial: Membuat WebSocket API dengan AWS integrasi](#)

Tutorial: Buat aplikasi WebSocket obrolan dengan WebSocket API, Lambda, dan DynamoDB

Dalam tutorial ini, Anda akan membuat aplikasi obrolan tanpa server dengan WebSocket API. Dengan WebSocket API, Anda dapat mendukung komunikasi dua arah antar klien. Klien dapat menerima pesan tanpa harus melakukan polling untuk pembaruan.

Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya. Pertama, Anda akan menggunakan CloudFormation template untuk membuat fungsi Lambda yang akan menangani permintaan API, serta tabel DynamoDB yang menyimpan klien Anda. IDs Kemudian, Anda akan menggunakan konsol API Gateway untuk membuat WebSocket API yang terintegrasi dengan fungsi Lambda Anda. Terakhir, Anda akan menguji API Anda untuk memverifikasi bahwa pesan dikirim dan diterima.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).

Anda juga `wscat` perlu terhubung ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).

Topik

- [Langkah 1: Buat fungsi Lambda dan tabel DynamoDB](#)

- [Langkah 2: Buat WebSocket API](#)
- [Langkah 3: Uji API Anda](#)
- [Langkah 4: Membersihkan](#)
- [Langkah selanjutnya: Otomatisasi dengan CloudFormation](#)

Langkah 1: Buat fungsi Lambda dan tabel DynamoDB

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#). Anda akan menggunakan template ini untuk membuat tabel Amazon DynamoDB untuk menyimpan klien aplikasi Anda. IDs Setiap klien yang terhubung memiliki ID unik yang akan kita gunakan sebagai kunci partisi tabel. Template ini juga menciptakan fungsi Lambda yang memperbarui koneksi klien Anda di DynamoDB dan menangani pengiriman pesan ke klien yang terhubung.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **websocket-api-chat-app-tutorial** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat WebSocket API

Anda akan membuat WebSocket API untuk menangani koneksi klien dan merutekan permintaan ke fungsi Lambda yang Anda buat di Langkah 1.

Untuk membuat WebSocket API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API. Kemudian untuk WebSocket API, pilih Build.
3. Untuk nama API, masukkan **websocket-chat-app-tutorial**.
4. Untuk jenis alamat IP, pilih IPv4.
5. Untuk ekspresi pemilihan Rute, masukkan **request.body.action**. Ekspresi pemilihan rute menentukan rute yang dipanggil API Gateway saat klien mengirim pesan.
6. Pilih Berikutnya.
7. Untuk rute Predefined, pilih Add \$connect, Add \$disconnect, dan Add \$default. Rute \$connect dan \$disconnect adalah rute khusus yang dipanggil API Gateway secara otomatis saat klien terhubung atau terputus dari API. API Gateway memanggil \$default rute ketika tidak ada rute lain yang cocok dengan permintaan.
8. Untuk rute Kustom, pilih Tambahkan rute khusus. Untuk kunci Rute, masukkan **sendMessage**. Rute kustom ini menangani pesan yang dikirim ke klien yang terhubung.
9. Pilih Berikutnya.
10. Di bawah Lampirkan integrasi, untuk setiap rute dan jenis Integrasi, pilih Lambda.

Untuk Lambda, pilih fungsi Lambda yang sesuai yang Anda buat di Langkah CloudFormation

1. Setiap nama fungsi cocok dengan rute. Misalnya, untuk rute \$connect, pilih fungsi bernama **websocket-chat-app-tutorial-ConnectHandler**.
11. Tinjau tahap yang dibuat API Gateway untuk Anda. Secara default, API Gateway membuat nama panggung `production` dan secara otomatis menerapkan API Anda ke tahap itu. Pilih Berikutnya.
12. Pilih Buat dan terapkan.

Langkah 3: Uji API Anda

Selanjutnya, Anda akan menguji API Anda untuk memastikan bahwa itu berfungsi dengan benar. Gunakan `wscat` perintah untuk terhubung ke API.

Untuk mendapatkan URL pemanggilan untuk API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pilih API Anda.
3. Pilih Tahapan, lalu pilih produksi.
4. Perhatikan WebSocket URL API Anda. URL akan terlihat seperti `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Untuk terhubung ke API Anda

1. Gunakan perintah berikut untuk terhubung ke API Anda. Saat Anda terhubung ke API, API Gateway memanggil `$connect` rute. Ketika rute ini dipanggil, ia memanggil fungsi Lambda yang menyimpan ID koneksi Anda di DynamoDB.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Buka terminal baru dan jalankan `wscat` perintah lagi dengan parameter berikut.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

Ini memberi Anda dua klien terhubung yang dapat bertukar pesan.

Untuk mengirim pesan

- API Gateway menentukan rute mana yang akan dipanggil berdasarkan ekspresi pemilihan rute API Anda. Ekspresi pemilihan rute API Anda adalah `$request.body.action`. Akibatnya, API Gateway memanggil `sendmessage` rute saat Anda mengirim pesan berikut:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

Fungsi Lambda yang terkait dengan rute yang dipanggil mengumpulkan klien dari DynamoDB. IDs Kemudian, fungsi tersebut memanggil API Gateway Management API dan mengirimkan pesan ke klien tersebut. Semua klien yang terhubung menerima pesan berikut:

```
< hello, everyone!
```


Untuk memanggil rute \$default API Anda

- API Gateway memanggil rute default API Anda saat klien mengirim pesan yang tidak cocok dengan rute yang Anda tentukan. Fungsi Lambda yang terkait dengan \$default rute menggunakan API Gateway Management API untuk mengirim informasi klien tentang koneksi mereka.

```
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
```

Untuk memutuskan sambungan dari API Anda

- Tekan **CTRL+C** untuk memutuskan sambungan dari API Anda. Saat klien terputus dari API Anda, API Gateway akan memanggil rute API Anda. \$disconnect Integrasi Lambda untuk \$disconnect rute API Anda menghapus ID koneksi dari DynamoDB.

Langkah 4: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus CloudFormation tumpukan dan WebSocket API Anda.

Untuk menghapus WebSocket API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada APIshalaman, pilih websocket-chat-app-tutorial API Anda. Pilih Tindakan, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

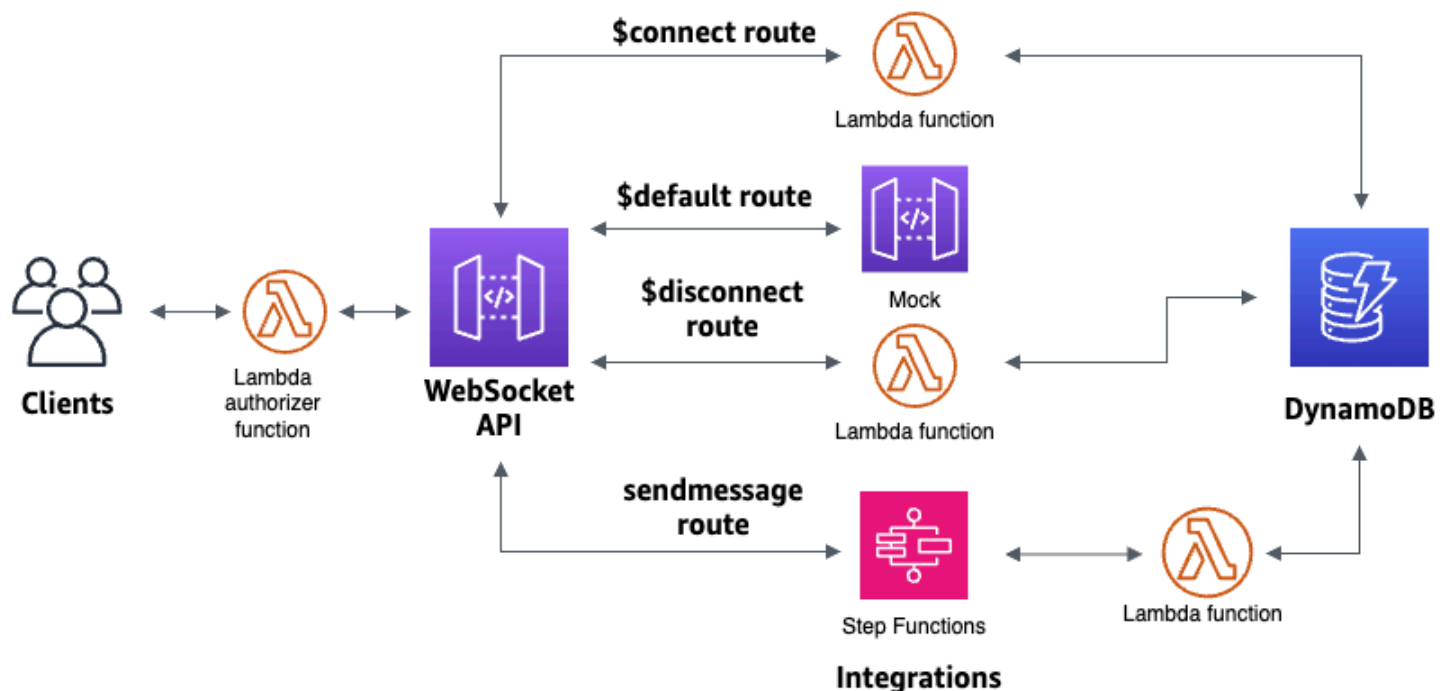
Langkah selanjutnya: Otomatisasi dengan CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua sumber AWS daya yang terlibat dalam tutorial ini. Untuk CloudFormation template yang membuat API ini dan semua resource terkait, lihat ws-chat-app.yaml.

Tutorial: Membuat WebSocket API dengan AWS integrasi

Dalam tutorial ini, Anda membuat aplikasi siaran tanpa server dengan WebSocket API. Klien dapat menerima pesan tanpa harus melakukan polling untuk pembaruan.

Tutorial ini menunjukkan cara menyiarkan pesan ke klien yang terhubung dan menyertakan contoh otorisasi Lambda, integrasi tiruan, dan integrasi non-proxy ke Step Functions.



Setelah membuat resource menggunakan CloudFormation template, Anda akan menggunakan konsol API Gateway untuk membuat WebSocket API yang terintegrasi dengan AWS resource Anda. Anda akan melampirkan otorisasi Lambda ke API Anda dan membuat integrasi AWS layanan dengan Step Functions untuk memulai eksekusi mesin status. Mesin status Step Functions akan memanggil fungsi Lambda yang mengirim pesan ke semua klien yang terhubung.

Setelah membangun API, Anda akan menguji koneksi ke API dan memverifikasi bahwa pesan dikirim dan diterima. Tutorial ini membutuhkan waktu sekitar 45 menit untuk menyelesaikannya.

Topik

- [Prasyarat](#)
- [Langkah 1: Buat sumber daya](#)
- [Langkah 2: Buat WebSocket API](#)
- [Langkah 3: Buat Authorizer Lambda](#)
- [Langkah 4: Buat integrasi dua arah tiruan](#)
- [Langkah 5: Buat integrasi non-proxy dengan Step Functions](#)
- [Langkah 6: Uji API Anda](#)
- [Langkah 7: Bersihkan](#)
- [Langkah selanjutnya](#)

Prasyarat

Anda membutuhkan prasyarat berikut:

- AWS Akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat [Prasyarat](#).
- `wscat` untuk terhubung ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Gunakan wscat untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).

Kami menyarankan Anda menyelesaikan tutorial aplikasi WebSocket obrolan sebelum memulai tutorial ini. Untuk menyelesaikan tutorial aplikasi WebSocket obrolan, lihat [the section called “Tutorial: Membuat aplikasi WebSocket obrolan”](#).

Langkah 1: Buat sumber daya

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#). Anda akan menggunakan template ini untuk membuat yang berikut:

- Fungsi Lambda yang menangani permintaan API dan mengotorisasi akses ke API Anda.
- Tabel DynamoDB untuk menyimpan IDs klien dan identifikasi pengguna utama yang dikembalikan oleh otorisasi Lambda.
- Mesin status Step Functions untuk mengirim pesan ke klien yang terhubung.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **websocket-step-functions-tutorial** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Pilih tab Output untuk melihat sumber daya yang Anda buat dan mereka ARNs. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat WebSocket API

Anda akan membuat WebSocket API untuk menangani koneksi klien dan merutekan permintaan ke sumber daya yang Anda buat di Langkah 1.

Untuk membuat WebSocket API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API. Kemudian untuk WebSocket API, pilih Build.
3. Untuk nama API, masukkan **websocket-step-functions-tutorial**.
4. Untuk jenis alamat IP, pilih IPv4.
5. Untuk ekspresi pemilihan Rute, masukkan **request.body.action**.

Ekspresi pemilihan rute menentukan rute yang dipanggil API Gateway saat klien mengirim pesan.

6. Pilih Berikutnya.

7. Untuk rute Predefined, pilih Add \$connect, Add \$disconnect, Add \$default.

Rute \$connect dan \$disconnect adalah rute khusus yang dipanggil API Gateway secara otomatis saat klien terhubung atau terputus dari API. API Gateway memanggil rute \$default ketika tidak ada rute lain yang cocok dengan permintaan. Anda akan membuat rute khusus untuk terhubung ke Step Functions setelah Anda membuat API Anda.

8. Pilih Berikutnya.

9. Untuk Integrasi untuk \$connect, lakukan hal berikut:

- a. Untuk jenis Integrasi, pilih Lambda.
- b. Untuk fungsi Lambda, pilih fungsi Lambda \$connect yang sesuai yang Anda buat CloudFormation di Langkah 1. Nama fungsi Lambda harus dimulai dengan. **websocket-step**

10. Untuk Integrasi untuk \$disconnect, lakukan hal berikut:

- a. Untuk jenis Integrasi, pilih Lambda.
- b. Untuk fungsi Lambda, pilih fungsi Lambda \$disconnect yang sesuai yang Anda buat di Langkah 1. CloudFormation Nama fungsi Lambda harus dimulai dengan. **websocket-step**

11. Untuk Integrasi untuk \$ default, pilih mock.

Dalam integrasi tiruan, API Gateway mengelola respons rute tanpa backend integrasi.

12. Pilih Berikutnya.

13. Tinjau tahap yang dibuat API Gateway untuk Anda. Secara default, API Gateway membuat tahap bernama production dan secara otomatis menerapkan API Anda ke tahap itu. Pilih Berikutnya.

14. Pilih Buat dan terapkan.

Langkah 3: Buat Authorizer Lambda

Untuk mengontrol akses ke WebSocket API Anda, Anda membuat otorisasi Lambda. CloudFormation Template membuat fungsi otorisasi Lambda untuk Anda. Anda dapat melihat fungsi Lambda di konsol Lambda. Nama harus dimulai dengan **websocket-step-functions-tutorial-AuthorizerHandler**. Fungsi Lambda ini menolak semua panggilan ke WebSocket API kecuali headernya. Authorization Allow Fungsi Lambda juga meneruskan \$context.authorizer.principalId variabel ke API Anda, yang nantinya digunakan dalam tabel DynamoDB untuk mengidentifikasi pemanggil API.

Pada langkah ini, Anda mengonfigurasi rute \$connect untuk menggunakan Lambda Authorizer.

Untuk membuat otorisasi Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk nama Authorizer, masukkan **LambdaAuthorizer**.
5. Untuk Authorizer ARN, masukkan nama otorisasi yang dibuat oleh template.
CloudFormation Nama harus dimulai dengan **websocket-step-functions-tutorial-AuthorizerHandler**.

 Note

Kami menyarankan Anda untuk tidak menggunakan contoh otorisasi ini untuk produksi APIs Anda.

6. Untuk tipe sumber Identity, pilih Header. Untuk Kunci, masukkan **Authorization**.
7. Pilih Buat Authorizer.

Setelah Anda membuat authorizer Anda, Anda melampirkannya ke \$connect rute API Anda.

Untuk melampirkan otorisasi ke rute \$connect

1. Di panel navigasi utama, pilih Rute.
2. Pilih rute \$connect.
3. Di bagian Pengaturan permintaan rute, pilih Edit.
4. Untuk Otorisasi, pilih menu tarik-turun, lalu pilih otorisasi permintaan Anda.
5. Pilih Simpan perubahan.

Langkah 4: Buat integrasi dua arah tiruan

Selanjutnya, Anda membuat integrasi tiruan dua arah untuk rute \$default. Integrasi tiruan memungkinkan Anda mengirim respons ke klien tanpa menggunakan backend. Saat Anda membuat integrasi untuk rute \$default, Anda dapat menunjukkan kepada klien cara berinteraksi dengan API Anda.

Anda mengonfigurasi rute \$default untuk menginformasikan klien agar menggunakan rute sendmessage.

Untuk membuat integrasi tiruan

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih rute \$ default, lalu pilih tab Permintaan integrasi.
3. Untuk templat Permintaan, pilih Edit.
4. Untuk ekspresi pemilihan Template, masukkan **200**, lalu pilih Edit.
5. Pada tab Permintaan integrasi, untuk templat Permintaan, pilih Buat templat.
6. Untuk kunci Template, masukkan **200**.
7. Untuk menghasilkan template, masukkan template pemetaan berikut:

```
{"statusCode": 200}
```

Pilih Buat templat.

Hasilnya akan terlihat seperti berikut ini:

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. Pada panel rute \$default, pilih Aktifkan komunikasi dua arah.
9. Pilih tab Integration response, lalu pilih Create Integration Response.
10. Untuk tombol Respons, masukkan **\$default**.
11. Untuk ekspresi pemilihan Template, masukkan **200**.
12. Pilih Buat respons.
13. Di bawah Templat respons, pilih Buat templat.
14. Untuk kunci Template, masukkan **200**.
15. Untuk template Response, masukkan template pemetaan berikut:

```

{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}

```

16. Pilih Buat templat.

Hasilnya akan terlihat seperti berikut ini:

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 EditDelete Template selection expression 200 Response templates Create template 200 EditDelete <pre>1 {"Use the sendmessage route to send a message. Connection ID: 2 \$context.connectionId"} 3</pre>			

Langkah 5: Buat integrasi non-proxy dengan Step Functions

Selanjutnya, Anda membuat rute sendmessage. Klien dapat memanggil rute sendmessage untuk menyiarkan pesan ke semua klien yang terhubung. Rute sendmessage memiliki integrasi AWS layanan non-proxy dengan. AWS Step Functions Integrasi memanggil [StartExecution](#) perintah untuk mesin status Step Functions yang dibuat CloudFormation template untuk Anda.

Untuk membuat integrasi non-proxy

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat rute.

3. Untuk kunci Rute, masukkan **sendMessage**.
4. Untuk jenis Integrasi, pilih AWS layanan.
5. Untuk AWS Wilayah, masukkan Wilayah tempat Anda menerapkan CloudFormation template Anda.
6. Untuk AWS layanan, pilih Step Functions.
7. Untuk metode HTTP, pilih POST.
8. Untuk nama Action, masukkan **StartExecution**.
9. Untuk peran Eksekusi, masukkan peran eksekusi yang dibuat oleh CloudFormation template. Namanya seharusnya **WebSocketTutorialApiRole**.
10. Pilih Buat rute.

Selanjutnya, Anda membuat template pemetaan untuk mengirim parameter permintaan ke mesin status Step Functions.

Untuk membuat template pemetaan

1. Pilih rute **sendMessage**, lalu pilih tab **Permintaan integrasi**.
2. Di bagian **Permintaan template**, pilih **Edit**.
3. Untuk ekspresi pemilihan Template, masukkan **\\$default**.
4. Pilih **Edit**.
5. Di bagian **Permintaan template**, pilih **Buat template**.
6. Untuk kunci Template, masukkan **\\$default**.
7. Untuk menghasilkan template, masukkan template pemetaan berikut:

```
#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"
}
```

Ganti *stateMachineArn* dengan ARN dari mesin negara yang dibuat oleh CloudFormation

Template pemetaan melakukan hal berikut:

- Menciptakan variabel `$domain` menggunakan variabel konteks `domainName`.
- Menciptakan variabel `$stage` menggunakan variabel konteks `stage`.

`$stage` Variabel `$domain` dan diperlukan untuk membangun URL callback.

- Mengambil pesan `sendMessage` JSON yang masuk, dan mengekstrak properti `message`
- Menciptakan input untuk mesin negara. Input adalah domain dan tahap WebSocket API dan pesan dari `sendMessage` rute.

8. Pilih Buat templat.

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  }
```

Anda dapat membuat integrasi non-proxy pada rute \$connect atau \$disconnect, untuk langsung menambahkan atau menghapus ID koneksi dari tabel DynamoDB, tanpa menjalankan fungsi Lambda.

Langkah 6: Uji API Anda

Selanjutnya, Anda akan menerapkan dan menguji API Anda untuk memastikannya berfungsi dengan benar. Anda akan menggunakan `wscat` perintah untuk terhubung ke API dan kemudian, Anda akan menggunakan perintah garis miring untuk mengirim bingkai ping untuk memeriksa koneksi ke WebSocket API.

Untuk men-deploy API Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Rute.
3. Pilih Deploy API.
4. Untuk Panggung, pilih produksi.
5. (Opsional) Untuk deskripsi Deployment, masukkan deskripsi.
6. Pilih Deploy.

Setelah menerapkan API, Anda dapat memanggilnya. Gunakan URL pemanggilan untuk memanggil API Anda.

Untuk mendapatkan URL pemanggilan untuk API Anda

1. Pilih API Anda.
2. Pilih Tahapan, lalu pilih produksi.
3. Perhatikan WebSocket URL API Anda. URL akan terlihat seperti `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Sekarang setelah Anda memiliki URL pemanggilan, Anda dapat menguji koneksi ke WebSocket API Anda.

Untuk menguji koneksi ke API

1. Gunakan perintah berikut untuk terhubung ke API Anda. Pertama, Anda menguji koneksi dengan memanggil `/ping` jalur.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H  
"Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Masukkan perintah berikut untuk melakukan ping pada bingkai kontrol. Anda dapat menggunakan bingkai kontrol untuk tujuan keepalive dari sisi klien.

```
/ping
```

Hasilnya akan terlihat seperti berikut ini:

```
< Received pong (data: "")
```

Sekarang setelah Anda menguji koneksi, Anda dapat menguji apakah API Anda berfungsi dengan benar. Pada langkah ini, Anda membuka jendela terminal baru sehingga WebSocket API dapat mengirim pesan ke semua klien yang terhubung.

Untuk menguji API Anda

1. Buka terminal baru dan jalankan wscat perintah lagi dengan parameter berikut.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H  
"Authorization: Allow"
```

```
Connected (press CTRL+C to quit)
```

2. API Gateway menentukan rute mana yang akan dipanggil berdasarkan ekspresi pemilihan permintaan rute API Anda. Ekspresi pemilihan rute API Anda adalah `$request.body.action`. Akibatnya, API Gateway memanggil `sendmessage` rute saat Anda mengirim pesan berikut:

```
{"action": "sendmessage", "message": "hello, from Step Functions!"}
```

Mesin status Step Functions yang terkait dengan rute memanggil fungsi Lambda dengan pesan dan URL callback. Fungsi Lambda memanggil API Gateway Management API dan mengirimkan pesan ke semua klien yang terhubung. Semua klien menerima pesan berikut:

```
< hello, from Step Functions!
```

Sekarang setelah Anda menguji WebSocket API Anda, Anda dapat memutuskan sambungan dari API Anda.

Untuk memutuskan sambungan dari API

- Tekan CTRL+C untuk memutuskan sambungan dari API Anda.

Saat klien terputus dari API Anda, API Gateway akan memanggil rute \$disconnect API Anda. Integrasi Lambda untuk rute \$disconnect API Anda menghapus ID koneksi dari DynamoDB.

Langkah 7: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus CloudFormation tumpukan dan WebSocket API Anda.

Untuk menghapus WebSocket API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada APIshalaman, pilih websocket-api Anda.
3. Pilih Tindakan, pilih Hapus, lalu konfirmasi pilihan Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Langkah selanjutnya

Anda dapat mengotomatiskan pembuatan dan pembersihan semua sumber AWS daya yang terlibat dalam tutorial ini. Untuk contoh CloudFormation template yang mengotomatiskan tindakan ini untuk tutorial ini, lihat [ws-sfn.zip](#).

API Gateway REST APIs

REST API di API Gateway adalah kumpulan sumber daya dan metode yang terintegrasi dengan titik akhir HTTP backend, fungsi Lambda, atau layanan lainnya. AWS Anda dapat menggunakan fitur API Gateway untuk membantu Anda dengan semua aspek siklus hidup API, mulai dari pembuatan hingga memantau produksi Anda. APIs

API Gateway REST APIs menggunakan model permintaan/respons di mana klien mengirim permintaan ke layanan dan layanan merespons kembali secara sinkron. Model semacam ini cocok untuk berbagai jenis aplikasi yang bergantung pada komunikasi sinkron.

Topik

- [Kembangkan REST API di API Gateway](#)
- [Publikasikan REST APIs agar pelanggan dapat memanggil](#)
- [Optimalkan kinerja REST APIs](#)
- [Mendistribusikan REST Anda APIs ke klien di API Gateway](#)
- [Lindungi REST Anda APIs di API Gateway](#)
- [Pantau REST APIs di API Gateway](#)

Kembangkan REST API di API Gateway

[Di Amazon API Gateway, Anda membuat REST API sebagai kumpulan entitas yang dapat diprogram yang dikenal sebagai sumber daya API Gateway.](#) Misalnya, Anda menggunakan [RestApi](#) sumber daya untuk mewakili API yang dapat berisi kumpulan entitas [Sumber Daya](#).

Setiap Resource entitas dapat memiliki satu atau lebih sumber daya [Metode](#). A Method adalah permintaan masuk yang dikirimkan oleh klien dan dapat berisi parameter permintaan berikut: parameter jalur, header, atau parameter string kueri. Selain itu, tergantung pada metode HTTP, permintaan dapat berisi badan. Metode Anda mendefinisikan bagaimana klien mengakses ekspos. Resource [Untuk mengintegrasikan Method dengan titik akhir backend, juga dikenal sebagai titik akhir integrasi, Anda membuat sumber daya Integrasi.](#) Ini meneruskan permintaan masuk ke URI titik akhir integrasi tertentu. Jika perlu, Anda dapat mengubah parameter permintaan atau badan permintaan untuk memenuhi persyaratan backend, atau Anda dapat membuat integrasi proxy, di mana API Gateway mengirimkan seluruh permintaan dalam format standar ke URI endpoint integrasi dan kemudian langsung mengirimkan respons ke klien.

Untuk tanggapan, Anda dapat membuat [MethodResponse](#) sumber daya untuk mewakili respons yang diterima oleh klien dan Anda membuat [IntegrationResponses](#) sumber daya untuk mewakili respons yang dikembalikan oleh backend. Gunakan respons integrasi untuk mengubah data respons backend sebelum mengembalikan data ke klien atau meneruskan respons backend apa adanya ke klien.

Contoh sumber daya untuk REST API

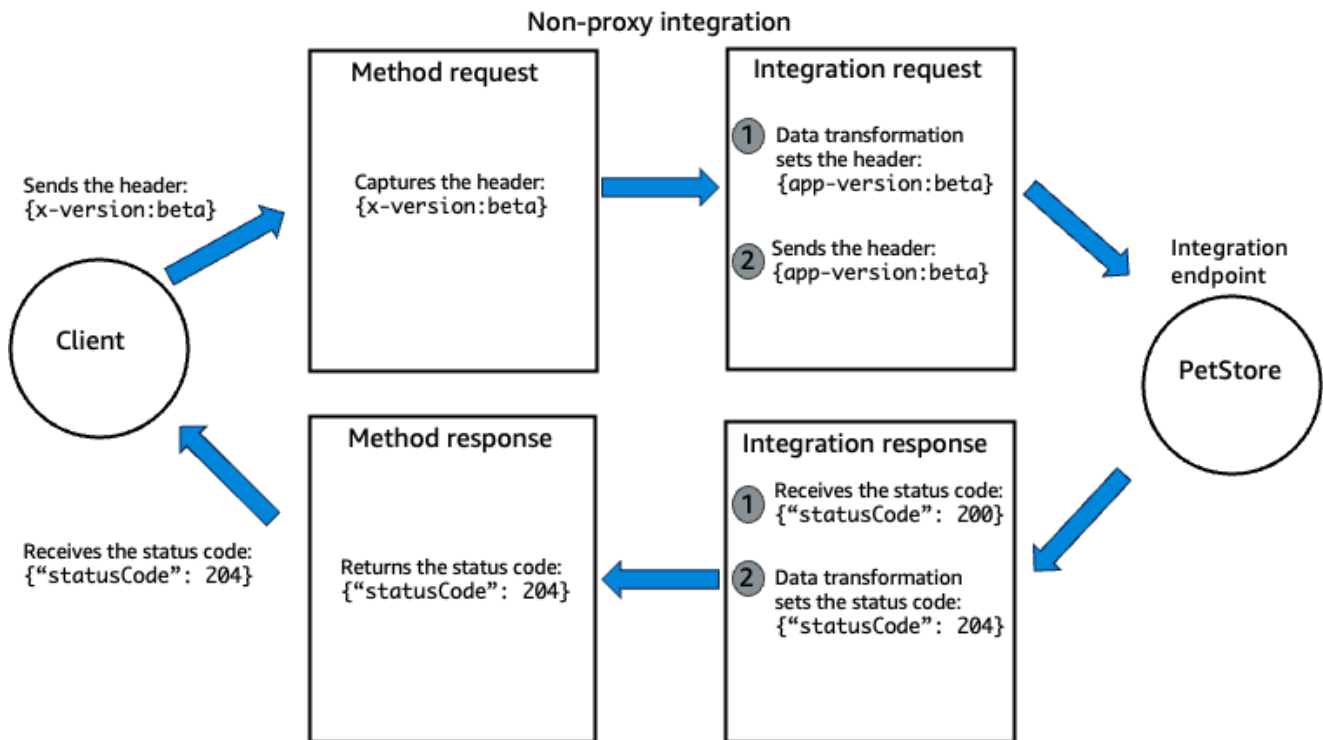
Diagram berikut menunjukkan bagaimana API Gateway mengimplementasikan request/response model ini untuk proxy HTTP dan integrasi non-proxy HTTP untuk sumber daya. GET /pets Klien mengirimkan `x-version:beta` header ke API Gateway dan API Gateway mengirimkan kode 204 status ke klien.

Dalam integrasi non-proxy, API Gateway melakukan transformasi data untuk memenuhi persyaratan backend, dengan memodifikasi permintaan integrasi dan respons integrasi. Dalam integrasi non-proxy, Anda dapat mengakses isi dalam permintaan metode tetapi Anda mengubahnya dalam permintaan integrasi. Ketika titik akhir integrasi mengembalikan respons dengan badan, Anda mengakses dan mengubahnya dalam respons integrasi. Anda tidak dapat memodifikasi tubuh dalam respons metode.

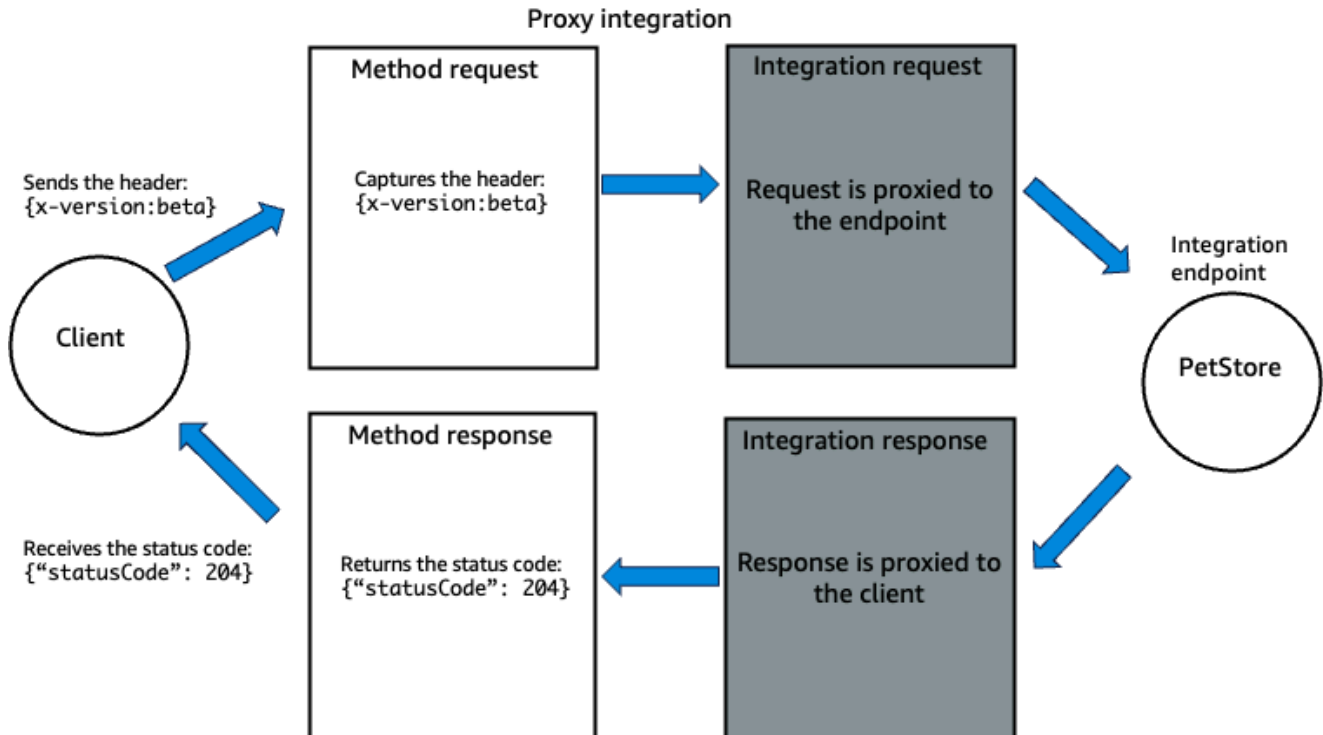
Dalam integrasi proxy, titik akhir integrasi memodifikasi permintaan dan respons. API Gateway tidak mengubah permintaan integrasi atau respons integrasi, dan mengirimkan permintaan masuk ke backend apa adanya.

Terlepas dari jenis integrasi, klien mengirim permintaan ke API Gateway dan API Gateway merespons secara sinkron.

Non-proxy integration



Proxy integration



Contoh log eksekusi berikut menunjukkan API Gateway apa yang akan masuk ke dalam contoh sebelumnya. Untuk kejelasan, beberapa nilai dan log awal telah dihapus:

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
```

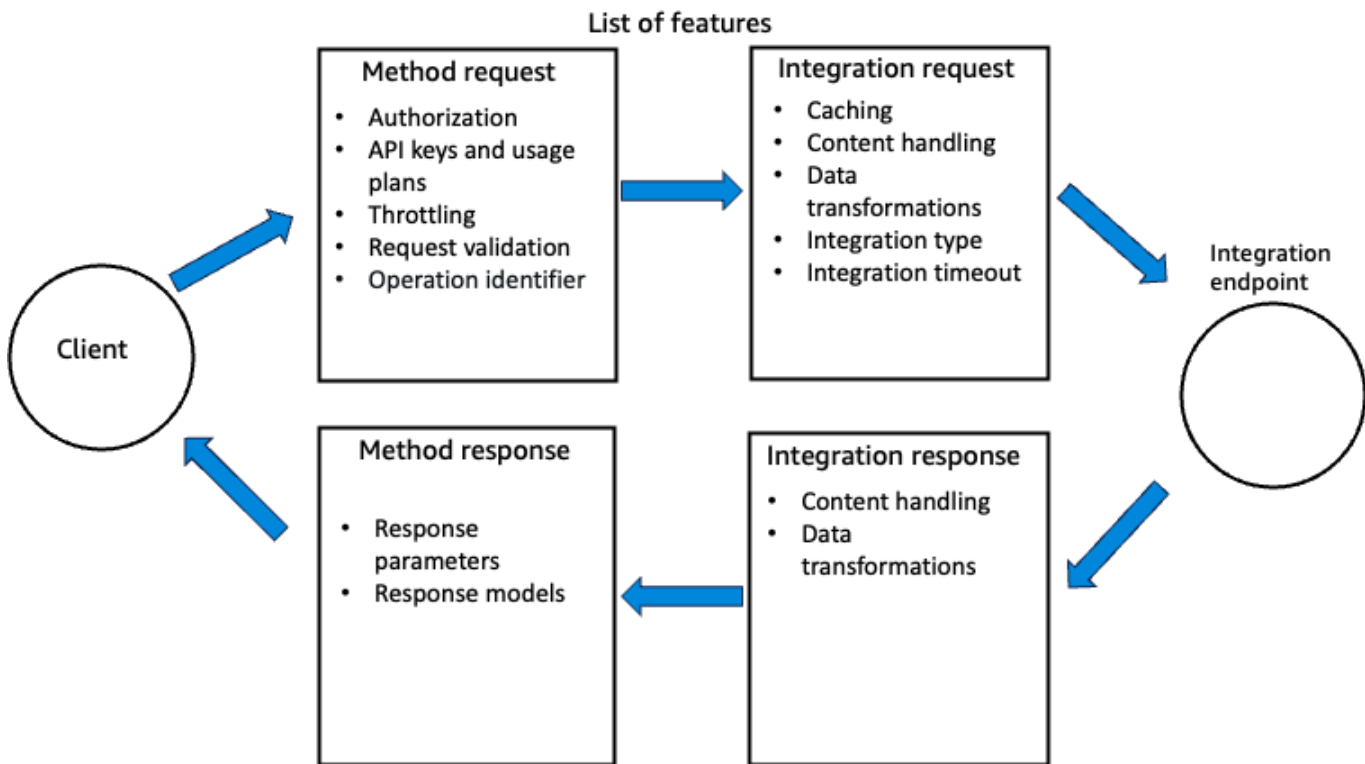
Untuk mengimpor API serupa dan mengujinya di API Konsol Manajemen AWS, lihat [contoh API](#).

Fitur REST API tambahan untuk pengembangan

API Gateway mendukung fitur tambahan untuk pengembangan REST API Anda. Misalnya, untuk membantu pelanggan memahami API Anda, Anda dapat memberikan dokumentasi untuk API. Untuk mengaktifkannya, tambahkan [DocumentationPart](#) resource untuk entitas API yang didukung.

[Untuk mengontrol cara klien memanggil API, gunakan izin IAM, otorisasi Lambda, atau kumpulan pengguna Amazon Cognito](#). Untuk mengukur penggunaan API Anda, siapkan [rencana penggunaan](#) untuk membatasi permintaan API. Anda dapat mengaktifkan ini saat membuat atau memperbarui API Anda.

Diagram berikut menunjukkan fitur yang tersedia untuk pengembangan REST API dan di mana dalam request/response model fitur ini dikonfigurasi.



Untuk pengantar tentang cara membuat API, lihat [the section called “Tutorial: Buat REST API dengan integrasi proxy Lambda”](#). Untuk mempelajari informasi selengkapnya tentang kemampuan API Gateway yang mungkin Anda gunakan saat mengembangkan REST API, lihat topik berikut. Topik ini berisi informasi konseptual dan prosedur yang dapat Anda lakukan menggunakan konsol API Gateway, API Gateway REST API AWS CLI, atau salah satu AWS SDK.

Topik

- [Jenis titik akhir API untuk REST APIs di API Gateway](#)
- [Kebijakan keamanan untuk REST API di API Gateway](#)
- [Jenis alamat IP untuk REST APIs di API Gateway](#)
- [Metode untuk REST APIs di API Gateway](#)
- [Kontrol dan kelola akses ke REST APIs di API Gateway](#)
- [Integrasi untuk REST APIs di API Gateway](#)
- [Minta validasi untuk REST APIs di API Gateway](#)
- [Transformasi data untuk REST API di API Gateway](#)
- [Tanggapan gateway untuk REST APIs di API Gateway](#)
- [CORS untuk REST APIs di API Gateway](#)

- [Jenis media biner untuk REST APIs di API Gateway](#)
- [Panggil REST APIs di API Gateway](#)
- [Kembangkan REST API menggunakan OpenAPI di API Gateway](#)

Jenis titik akhir API untuk REST APIs di API Gateway

Jenis [titik akhir API](#) mengacu pada nama host API. Jenis titik akhir API dapat dioptimalkan secara tepi, Regional, atau pribadi, tergantung dari mana sebagian besar lalu lintas API Anda berasal.

Titik akhir API yang dioptimalkan tepi

[Titik akhir API yang dioptimalkan tepi](#) biasanya merutekan permintaan ke CloudFront Point of Presence (POP) terdekat, yang dapat membantu jika klien Anda didistribusikan secara geografis. Ini adalah tipe endpoint default untuk API Gateway REST APIs.

Edge dioptimalkan APIs menggunakan huruf besar pada nama [header HTTP \(misalnya,\)](#). Cookie

CloudFront mengurutkan cookie HTTP dalam urutan alami dengan nama cookie sebelum meneruskan permintaan ke asal Anda. Untuk informasi selengkapnya tentang cara CloudFront memproses cookie, lihat [Caching Konten Berdasarkan Cookie](#).

Nama domain kustom apa pun yang Anda gunakan untuk API yang dioptimalkan tepi berlaku di semua wilayah.

Titik akhir API regional

[Titik akhir API Regional](#) ditujukan untuk klien di Wilayah yang sama. Ketika klien yang berjalan pada instans EC2 memanggil API di Wilayah yang sama, atau ketika API dimaksudkan untuk melayani sejumlah kecil klien dengan permintaan tinggi, API Regional mengurangi overhead koneksi.

Untuk API Regional, nama domain kustom apa pun yang Anda gunakan khusus untuk Wilayah tempat API diterapkan. Jika Anda menerapkan API Regional di beberapa Wilayah, API tersebut dapat memiliki nama domain kustom yang sama di semua Wilayah. Anda dapat menggunakan domain khusus bersama dengan Amazon Route 53 untuk melakukan tugas seperti perutean [berbasis latensi](#). Untuk informasi selengkapnya, lihat [the section called “Siapkan nama domain kustom Regional”](#) dan [the section called “Siapkan nama domain kustom yang dioptimalkan tepi di API Gateway”](#).

Titik akhir API regional meneruskan semua nama header melalui apa adanya.

 Note

Dalam kasus di mana klien API tersebar secara geografis, mungkin masih masuk akal untuk menggunakan titik akhir API Regional, bersama dengan CloudFront distribusi Amazon Anda sendiri untuk memastikan bahwa API Gateway tidak mengaitkan API dengan distribusi yang dikendalikan layanan. CloudFront Untuk informasi selengkapnya tentang kasus penggunaan ini, lihat [Bagaimana cara mengatur API Gateway dengan CloudFront distribusi saya sendiri?](#).

Titik akhir API pribadi

[Titik akhir API pribadi adalah titik akhir](#) API yang hanya dapat diakses dari Amazon Virtual Private Cloud (VPC) menggunakan titik akhir VPC antarmuka, yang merupakan antarmuka jaringan titik akhir (ENI) yang Anda buat di VPC Anda. Untuk informasi selengkapnya, lihat [the section called “REST pribadi APIs”](#).

Titik akhir API pribadi meneruskan semua nama header melalui apa adanya.

Mengubah jenis titik akhir API publik atau pribadi di API Gateway

Mengubah tipe titik akhir API mengharuskan Anda memperbarui konfigurasi API. Anda dapat mengubah jenis API yang ada menggunakan konsol API Gateway, the AWS CLI, atau AWS SDK untuk API Gateway. Jenis titik akhir tidak dapat diubah lagi hingga perubahan saat ini selesai, tetapi API Anda akan tersedia.

Perubahan tipe endpoint berikut didukung:

- Dari yang dioptimalkan tepi ke Regional atau pribadi
- Dari Regional hingga yang dioptimalkan tepi atau pribadi
- Dari pribadi ke Regional

Anda tidak dapat mengubah API pribadi menjadi API yang dioptimalkan tepi.

Jika Anda mengubah API publik dari yang dioptimalkan tepi ke Regional atau sebaliknya, perhatikan bahwa API yang dioptimalkan tepi mungkin memiliki perilaku yang berbeda dari API Regional. Misalnya, API yang dioptimalkan tepi menghapus header. Content-MD5 Setiap nilai MD5 hash yang diteruskan ke backend dapat dinyatakan dalam parameter string permintaan atau properti tubuh. Namun, API Regional meneruskan header ini, meskipun mungkin memetakan ulang nama header ke

beberapa nama lain. Memahami perbedaan akan membantu Anda memutuskan cara memperbarui API yang dioptimalkan tepi ke API Regional atau dari API Regional ke API yang dioptimalkan tepi.

Topik

- [Menggunakan konsol API Gateway untuk mengubah jenis titik akhir API](#)
- [Gunakan AWS CLI untuk mengubah tipe titik akhir API](#)

Menggunakan konsol API Gateway untuk mengubah jenis titik akhir API

Untuk mengubah jenis titik akhir API API Anda, lakukan salah satu set langkah berikut:

Untuk mengonversi titik akhir publik dari Regional atau yang dioptimalkan tepi dan sebaliknya

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih pengaturan API.
4. Di bagian detail API, pilih Edit.
5. Untuk jenis endpoint API, pilih Edge-optimized atau Regional.
6. Pilih Simpan perubahan.
7. Menerapkan ulang API Anda sehingga perubahan akan berlaku.

Untuk mengonversi titik akhir pribadi ke titik akhir Regional

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Edit kebijakan sumber daya untuk API Anda untuk menghapus penyebutan VPCs atau titik akhir VPC sehingga panggilan API dari luar VPC Anda maupun di dalam VPC Anda akan berhasil.
4. Pilih pengaturan API.
5. Di bagian detail API, pilih Edit.
6. Untuk jenis titik akhir API, pilih Regional.
7. Pilih Simpan perubahan.
8. Hapus kebijakan sumber daya dari API Anda.
9. Menerapkan ulang API Anda sehingga perubahan akan berlaku.

Karena Anda memigrasikan tipe titik akhir dari pribadi ke Regional, API Gateway mengubah jenis alamat IP menjadi IPv4. Untuk informasi selengkapnya, lihat [the section called “Jenis alamat IP untuk REST APIs di API Gateway”](#).

Untuk mengonversi titik akhir Regional menjadi titik akhir pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Buat kebijakan sumber daya yang memberikan akses ke titik akhir VPC atau VPC Anda. Untuk informasi selengkapnya, lihat [???](#).
4. Pilih pengaturan API.
5. Di bagian detail API, pilih Edit.
6. Untuk jenis endpoint API, pilih Private.
7. (Opsional) Untuk titik akhir VPC IDs, pilih titik akhir VPC yang ingin Anda kaitkan dengan IDs API pribadi Anda.
8. Pilih Simpan perubahan.
9. Menerapkan ulang API Anda sehingga perubahan akan berlaku.

Karena Anda memigrasikan tipe endpoint dari Regional ke private, API Gateway mengubah tipe alamat IP menjadi dualstack. Untuk informasi selengkapnya, lihat [the section called “Jenis alamat IP untuk REST APIs di API Gateway”](#).

Gunakan AWS CLI untuk mengubah tipe titik akhir API

[update-rest-api](#) Perintah berikut memperbarui API yang dioptimalkan tepi ke API Regional:

```
aws apigateway update-rest-api \  
  --rest-api-id a1b2c3 \  
  --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

Respons yang berhasil memiliki kode status 200 OK dan muatan yang mirip dengan berikut ini:

```
{  
  "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"  
}
```

[update-rest-api](#) Perintah berikut memperbarui API Regional ke API yang dioptimalkan tepi:

```
aws apigateway update-rest-api \  
--rest-api-id a1b2c3 \  
--patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Karena [put-rest-api](#) untuk memperbarui definisi API, itu tidak berlaku untuk memperbarui jenis titik akhir API.

Kebijakan keamanan untuk REST API di API Gateway

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh API Gateway. Saat klien Anda membuat handshake TLS ke API atau nama domain kustom Anda, kebijakan keamanan memberlakukan versi TLS dan cipher suite yang diterima oleh API Gateway. Kebijakan keamanan melindungi API dan nama domain kustom Anda dari masalah keamanan jaringan seperti manipulasi dan penyadapan antara klien dan server.

API Gateway mendukung kebijakan keamanan lama dan kebijakan keamanan yang ditingkatkan. TLS_1_0 dan TLS_1_2 merupakan kebijakan keamanan warisan. Gunakan kebijakan keamanan ini untuk kompatibilitas mundur. Kebijakan apa pun yang dimulai `SecurityPolicy_` adalah kebijakan keamanan yang ditingkatkan. Gunakan kebijakan ini untuk beban kerja yang diatur, tata kelola lanjutan, atau untuk menggunakan kriptografi pascakuantum. Jika Anda menggunakan kebijakan keamanan yang ditingkatkan, Anda juga harus menetapkan mode akses titik akhir untuk tata kelola tambahan. Untuk informasi selengkapnya, lihat [the section called “Mode akses titik akhir”](#).

Bagaimana API Gateway menerapkan kebijakan keamanan

Contoh berikut menunjukkan bagaimana API Gateway menerapkan kebijakan keamanan menggunakan kebijakan `SecurityPolicy_TLS13_1_3_2025_09` keamanan sebagai contoh.

Kebijakan `SecurityPolicy_TLS13_1_3_2025_09` keamanan menerima lalu lintas TLS 1.3 dan menolak lalu lintas TLS 1.2 dan TLS 1.0. Untuk lalu lintas TLS 1.3, kebijakan keamanan menerima rangkaian sandi berikut:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

API Gateway tidak menerima cipher suite lainnya. Misalnya, kebijakan keamanan akan menolak lalu lintas TLS 1.3 yang menggunakan AES128-SHA cipher suite. Untuk informasi selengkapnya tentang versi dan cipher TLS yang didukung, lihat [the section called “Kebijakan keamanan yang didukung”](#)

Untuk memantau protokol TLS dan cipher klien yang digunakan untuk mengakses API Gateway Anda, Anda dapat menggunakan variabel `$context.tlsVersion` dan `$context.cipherSuite` konteks dalam log akses Anda. Untuk informasi selengkapnya, lihat [the section called “Memantau”](#).

Mode akses titik akhir

Mode akses titik akhir adalah parameter tambahan yang harus Anda tentukan untuk API REST atau nama domain kustom apa pun yang menggunakan kebijakan keamanan yang ditingkatkan yang dimulai dengan `SecurityPolicy_`. Anda melakukannya saat membuat sumber daya atau jika Anda mengubah kebijakan keamanan dari kebijakan lama menjadi kebijakan yang disempurnakan.

Saat mode akses titik akhir disetel ke `STRICT`, permintaan apa pun ke REST API atau nama domain kustom Anda harus melewati pemeriksaan berikut:

- Permintaan harus berasal dari jenis titik akhir API Gateway yang sama dengan sumber daya Anda. Permintaan ini bisa berasal dari titik akhir Regional, titik akhir yang dioptimalkan untuk edge, atau titik akhir privat.
- Jika Anda menggunakan endpoint Regional atau pribadi, API Gateway menggunakan pencocokan host SNI. Jika Anda menggunakan titik akhir yang dioptimalkan tepi, API Gateway sesuai dengan perlindungan fronting domain CloudFront. Untuk informasi selengkapnya, lihat [Domain fronting](#).

Jika salah satu dari kondisi ini tidak terpenuhi, API Gateway menolak permintaan tersebut. Kami menyarankan Anda menggunakan mode akses `STRICT` titik akhir jika memungkinkan.

Untuk memigrasikan API atau nama domain yang ada agar menggunakan mode akses titik akhir yang ketat, perbarui kebijakan keamanan Anda terlebih dahulu ke kebijakan keamanan yang disempurnakan dan tetapkan mode akses titik akhir. `BASIC` Setelah Anda memvalidasi lalu lintas dan log akses Anda, atur mode akses titik akhir ke `STRICT`. Saat Anda memigrasikan mode akses titik akhir dari `STRICT` ke `BASIC`, titik akhir Anda tidak akan tersedia selama sekitar 15 menit saat perubahan menyebar.

Anda tidak boleh menyetel mode akses titik akhir STRICT untuk arsitektur aplikasi tertentu dan sebagai gantinya mengatur mode akses titik akhir ke. BASIC Tabel berikut menunjukkan beberapa arsitektur aplikasi dan rekomendasi sehingga REST API atau nama domain kustom Anda dapat menggunakan mode akses STRICT endpoint.

Arsitektur	Migrasi yang disarankan
Menggunakan titik akhir VPC untuk mengakses nama domain kustom publik.	Arsitektur ini menggunakan lalu lintas tipe cross-endpoint. Kami menyarankan Anda untuk bermigrasi ke the section called “Nama domain khusus untuk pribadi APIs” .
Menggunakan metode apa pun untuk memanggil API pribadi yang tidak menggunakan nama domain khusus atau nama DNS pribadi.	Arsitektur ini menciptakan ketidakcocokan antara header host dan SNI yang digunakan dalam jabat tangan TLS dan tidak melewati CloudFront batasan fronting domain. Kami menyarankan Anda memigrasikan VPC Anda untuk menggunakan DNS pribadi.
Menggunakan domain sharding untuk mendistribusikan konten di beberapa domain atau subdomain.	Arsitektur ini menciptakan ketidakcocokan antara header host dan SNI yang digunakan dalam jabat tangan TLS dan tidak melewati CloudFront batasan fronting domain. Kami menyarankan Anda menggunakan HTTP/2 dan bermigrasi dari anti-pola ini.

Berikut ini adalah pertimbangan untuk menggunakan mode akses titik akhir:

- Jika mode akses titik akhir API atau nama domain adalah STRICT, Anda tidak dapat mengubah jenis titik akhir. Untuk mengubah tipe titik akhir, pertama-tama ubah mode akses titik akhir menjadi BASIC
- Setelah Anda mengubah mode akses titik akhir dari BASIC ke STRICT, ada penundaan 15 menit bagi API Gateway untuk menerapkan mode akses titik akhir yang ketat.
- Jika Anda mengubah kebijakan keamanan dari kebijakan yang dimulai dengan `SecurityPolicy_` kebijakan lama, Anda harus membatalkan pengaturan mode akses titik akhir.

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk kebijakan keamanan untuk REST API di API Gateway:

- Anda dapat mengimpor kebijakan keamanan dalam file definisi OpenAPI. Untuk informasi selengkapnya, lihat [x-amazon-apigateway-endpoint-modus akses](#).
- API Anda dapat dipetakan ke nama domain khusus dengan kebijakan keamanan yang berbeda dari API Anda. Saat Anda memanggil nama domain kustom tersebut, API Gateway menggunakan kebijakan keamanan API untuk menegosiasikan jabat tangan TLS. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.
- Jika Anda mengubah kebijakan keamanan Anda, dibutuhkan sekitar 15 menit untuk pembaruan selesai. Anda dapat memantau `apiStatus` API Anda. Saat API Anda diperbarui, `apiStatus` adalah `UPDATING` dan ketika selesai, itu akan menjadi `AVAILABLE`. Ketika status API Anda `UPDATING`, Anda masih dapat memanggilnya.
- API Gateway mendukung kebijakan keamanan pada semua API. Namun, Anda hanya dapat memilih kebijakan keamanan untuk REST API. API Gateway hanya mendukung kebijakan `TLS_1_2` keamanan untuk HTTP atau WebSocket API.
- Anda tidak dapat memperbarui kebijakan keamanan untuk API dari `TLS_1_0` ke `TLS_1_2`.
- Beberapa kebijakan keamanan mendukung rangkaian sandi ECDSA dan RSA. Jika Anda menggunakan jenis kebijakan ini dengan nama domain kustom, rangkaian sandi cocok dengan jenis kunci sertifikat yang disediakan pelanggan, baik RSA atau ECDSA. Jika Anda menggunakan jenis kebijakan ini dengan REST API, cipher suite cocok dengan cipher suite yang kompatibel dengan tipe sertifikat RSA.

Kebijakan keamanan yang didukung

Tabel berikut menjelaskan [kebijakan keamanan](#) yang dapat ditentukan untuk setiap jenis endpoint REST API dan jenis nama domain kustom. Kebijakan ini memungkinkan Anda untuk mengontrol koneksi masuk. API Gateway hanya mendukung TLS 1.2 saat keluar. Anda dapat memperbarui kebijakan keamanan untuk API atau nama domain kustom kapan saja.

Kebijakan yang terkandung FIPS dalam judul kompatibel dengan Federal Information Processing Standard (FIPS), yang merupakan standar pemerintah AS dan Kanada yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Untuk mempelajari lebih lanjut, lihat [Federal Information Processing Standard \(FIPS\) 140](#) di halaman Kepatuhan Keamanan AWS Cloud.

Semua kebijakan FIPS memanfaatkan modul kriptografi yang divalidasi AWS-LC FIPS. Untuk mempelajari lebih lanjut, lihat halaman [Modul AWS-LC Kriptografi](#) di situs Program Validasi Modul Kriptografi NIST.

Kebijakan yang terdapat PQ dalam judul menggunakan [Post-Quantum Cryptography \(PQC\)](#) untuk mengimplementasikan algoritma pertukaran kunci hibrida untuk TLS guna memastikan kerahasiaan lalu lintas terhadap ancaman komputasi kuantum masa depan.

Kebijakan yang terdapat PFS dalam judul menggunakan [Perfect Forward Secrecy \(PFS\)](#) untuk memastikan kunci sesi tidak dikompromikan.

Kebijakan yang berisi keduanya FIPS dan PQ dalam judulnya mendukung kedua fitur ini.

Kebijakan keamanan default

Saat Anda membuat REST API atau domain kustom baru, sumber daya akan ditetapkan kebijakan keamanan default. Tabel berikut menunjukkan kebijakan keamanan default untuk sumber daya ini.

Sumber Daya	Nama kebijakan keamanan default		
API Regional	TLS_1_0		
Edge-optimized API	TLS_1_0		
API Privat	TLS_1_2		
Domain regional	TLS_1_2		
Edge-optimized domain	TLS_1_2		
Domain privat	TLS_1_2		

Kebijakan keamanan yang didukung untuk API Regional dan pribadi serta nama domain khusus

Tabel berikut menjelaskan kebijakan keamanan yang dapat ditentukan untuk API Regional dan pribadi serta nama domain kustom:

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
SecurityPolicy_TLS13_1_3_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
		• ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Kebijakan keamanan yang didukung untuk API yang dioptimalkan tepi dan nama domain khusus

Tabel berikut menjelaskan kebijakan keamanan yang dapat ditentukan untuk API yang dioptimalkan tepi dan nama domain kustom yang dioptimalkan tepi:

Nama kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
SecurityPolicy_TLS13_2025_T EPI	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS 12_PFS_2025_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Nama kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
		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

Nama kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Nama kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
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

Nama kebijakan keamanan	Versi TLS yang didukung	Cipher yang didukung
		• ECDHE-RSA-AES256-SHA • AES256-SHA256 • AES256-SHA

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246 menggunakan nama yang berbeda untuk cipher yang sama. Tabel berikut memetakan nama OpenSSL ke nama RFC untuk setiap cipher. Untuk informasi selengkapnya, lihat [cipher](#) di Dokumentasi OpenSSL.

Nama sandi OpenSSL	Nama sandi RFC
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
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

Nama sandi OpenSSL	Nama sandi RFC
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

Cara mengubah kebijakan keamanan

Anda dapat mengubah kebijakan keamanan untuk API Anda. Jika Anda mengirim lalu lintas ke API Anda melalui nama domain kustom Anda, API dan nama domain kustom tidak perlu memiliki kebijakan keamanan yang sama. Saat Anda memanggil nama domain kustom tersebut, API Gateway menggunakan kebijakan keamanan API untuk menegosiasikan jabat tangan TLS. Namun, untuk konsistensi, kami menyarankan Anda menggunakan kebijakan keamanan yang sama untuk nama domain dan API kustom Anda.

Jika Anda mengubah kebijakan keamanan Anda, dibutuhkan sekitar 15 menit untuk pembaruan selesai. Anda dapat memantau `apiStatus` API Anda. Saat API Anda diperbarui, `apiStatus`

adalah UPDATING dan ketika selesai, itu akan terjadiAVAILABLE. Saat API Anda diperbarui, Anda masih dapat memanggilnya.

Konsol Manajemen AWS

Untuk mengubah kebijakan keamanan API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih pengaturan API, lalu pilih Edit.
4. Untuk kebijakan Keamanan, pilih kebijakan baru yang dimulai denganSecurityPolicy_.
5. Untuk mode akses Endpoint, pilih Strict.
6. Pilih Simpan perubahan.

Menerapkan ulang API Anda agar perubahan diterapkan. Karena Anda mengubah mode akses titik akhir menjadi ketat, akan memakan waktu sekitar 15 menit agar perubahan menyebar sepenuhnya.

AWS CLI

Perintah [update-rest-api](#) berikut memperbarui API untuk menggunakan kebijakan keamanan: 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"
    }
  ]'
```

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}
```

Perintah [update-rest-api](#) berikut [memperbarui API](#) yang menggunakan kebijakan keamanan yang disempurnakan untuk menggunakan kebijakan keamanan. 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": ""
    }
  ]'
```

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}
```

Jenis alamat IP untuk REST APIs di API Gateway

Saat membuat API, Anda menentukan jenis alamat IP yang dapat menjalankan API Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk menjalankan API Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat menjalankan API Anda. Kami menyarankan Anda mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Untuk membatasi API hanya IPv6 lalu lintas, Anda dapat membuat kebijakan sumber daya dan membatasi alamat IP sumber hanya IPv6 untuk rentang. Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi API. Perubahan ini akan segera berlaku, dan Anda tidak perlu menerapkan ulang API Anda. Untuk informasi selengkapnya, lihat [the section called “Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber”](#)

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda:

- Jenis alamat IP default untuk semua Regional dan yang dioptimalkan tepi APIs adalah. IPv4
- Private hanya APIs dapat memiliki tipe alamat IP dualstack.
- Jika Anda mengubah jenis alamat IP untuk API yang ada dari IPv4 ke dualstack, konfirmasi bahwa kebijakan apa pun yang mengontrol akses ke Anda APIs telah diperbarui untuk IPv6

memperhitungkan panggilan. Ketika Anda mengubah jenis alamat IP, perubahan akan segera berlaku.

- Jika Anda memigrasikan tipe titik akhir API dari Regional atau yang dioptimalkan tepi ke pribadi, API Gateway mengubah jenis alamat IP menjadi dualstack. Untuk informasi selengkapnya, lihat [the section called “Mengubah jenis titik akhir API publik atau pribadi”](#).
- Jika Anda memigrasikan tipe titik akhir API dari pribadi ke Regional, Anda harus menyetel jenis alamat IP ke dualstack. Setelah migrasi titik akhir selesai, Anda dapat mengubah jenis alamat IP menjadi IPv4. Untuk informasi selengkapnya, lihat [the section called “Mengubah jenis titik akhir API publik atau pribadi”](#).
- API Anda dapat dipetakan ke nama domain khusus dengan jenis alamat IP yang berbeda dari API Anda. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.
- Anda tidak dapat menggunakan file definisi eksternal untuk mengonfigurasi jenis alamat IP API Anda.

Mengubah jenis alamat IP dari REST API

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi API. Anda dapat memperbarui konfigurasi API dengan menggunakan Konsol Manajemen AWS, the AWS CLI CloudFormation, atau AWS SDK. Jika Anda mengubah jenis alamat IP API, Anda tidak menerapkan ulang API agar perubahan diterapkan. Sebelum Anda mengubah jenis alamat IP, konfirmasi bahwa kebijakan apa pun yang mengontrol akses ke Anda APIs telah diperbarui ke akun untuk IPv6 panggilan.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP dari REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih pengaturan API, lalu pilih Edit.
4. Untuk jenis alamat IP, pilih salah satu IPv4 atau Dualstack.
5. Pilih Simpan perubahan.

Perubahan pada konfigurasi API Anda akan segera berlaku.

AWS CLI

[update-rest-api](#) Perintah berikut memperbarui API untuk memiliki tipe alamat IP dualstack:

```
aws apigateway update-rest-api \  
  --rest-api-id abcd1234 \  
  --patch-operations "op='replace',path='/endpointConfiguration/  
ipAddressType',value='dualstack'"
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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"  
}
```

Metode untuk REST APIs di API Gateway

Di API Gateway, metode API mewujudkan [permintaan metode](#) dan [respons metode](#). Anda menyiapkan metode API untuk menentukan apa yang harus atau harus dilakukan klien untuk mengirimkan permintaan untuk mengakses layanan di backend dan untuk menentukan tanggapan yang diterima klien sebagai imbalannya. Untuk masukan, Anda dapat memilih parameter permintaan metode, atau muatan yang berlaku, agar klien dapat menyediakan data yang diperlukan atau opsional pada waktu berjalan. Untuk output, Anda menentukan kode status respons metode, header, dan badan yang berlaku sebagai target untuk memetakan data respons backend, sebelum dikembalikan ke klien. Untuk membantu pengembang klien memahami perilaku dan format input dan output API Anda, Anda dapat [mendokumentasikan](#) API Anda dan [memberikan pesan kesalahan yang tepat](#) untuk [permintaan yang tidak valid](#).

Permintaan metode API adalah permintaan HTTP. Untuk menyiapkan permintaan metode, Anda mengonfigurasi metode HTTP (atau kata kerja), jalur ke [sumber daya](#) API, header, parameter string kueri yang berlaku. Anda juga mengonfigurasi payload ketika metode HTTP adalah POST, PUT, atau PATCH. Misalnya, untuk mengambil hewan peliharaan menggunakan [API PetStore sampel](#), Anda menentukan permintaan metode API GET /pets/{petId}, di mana {petId} merupakan parameter jalur yang dapat mengambil nomor pada waktu proses.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Jika klien menentukan jalur yang salah, misalnya, /pet/1 atau /pets/one bukan /pets/1, pengecualian dilemparkan.

Respons metode API adalah respons HTTP dengan kode status yang diberikan. Untuk integrasi non-proxy, Anda harus menyiapkan respons metode untuk menentukan target pemetaan yang diperlukan atau opsional. Ini mengubah header atau badan respons integrasi ke header atau badan respons metode terkait. Pemetaan dapat sederhana [transformasi identitas](#) yang melewati header atau badan melalui integrasi apa adanya. Misalnya, respons 200 metode berikut menunjukkan contoh passthrough dari respons integrasi yang berhasil apa adanya.

```
200 OK
Content-Type: application/json
...

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

Pada prinsipnya, Anda dapat menentukan respons metode yang sesuai dengan respons tertentu dari backend. Biasanya, ini melibatkan respons 2XX, 4XX, dan 5XX. Namun, ini mungkin tidak praktis, karena seringkali Anda mungkin tidak tahu sebelumnya semua tanggapan yang mungkin dikembalikan oleh backend. Dalam praktiknya, Anda dapat menetapkan satu respons metode sebagai default untuk menangani respons yang tidak diketahui atau tidak dipetakan dari backend. Ini adalah praktik yang baik untuk menetapkan respons 500 sebagai default. Bagaimanapun, Anda harus menyiapkan setidaknya satu respons metode untuk integrasi non-proxy. Jika tidak, API

Gateway mengembalikan respons kesalahan 500 ke klien bahkan ketika permintaan berhasil di backend.

Untuk mendukung SDK yang diketik dengan kuat, seperti Java SDK, untuk API Anda, Anda harus menentukan model data untuk input untuk permintaan metode, dan menentukan model data untuk output dari respons metode.

Prasyarat

Sebelum menyiapkan metode API, verifikasi hal berikut:

- Anda harus memiliki metode yang tersedia di API Gateway. Ikuti petunjuk dalam [Tutorial: Membuat REST API dengan integrasi non-proxy HTTP](#).
- Jika Anda ingin metode berkomunikasi dengan fungsi Lambda, Anda harus sudah membuat peran pemanggilan Lambda dan peran eksekusi Lambda di IAM. Anda juga harus membuat fungsi Lambda yang dengannya metode Anda akan berkomunikasi. AWS Lambda Untuk membuat peran dan fungsi, gunakan instruksi dalam [Buat fungsi Lambda untuk integrasi non-proxy LambdaPilih tutorial AWS Lambda integrasi](#).
- Jika Anda ingin metode berkomunikasi dengan integrasi proxy HTTP atau HTTP, Anda harus sudah membuat, dan memiliki akses ke, URL titik akhir HTTP yang dengannya metode Anda akan berkomunikasi.
- Verifikasi bahwa sertifikat Anda untuk titik akhir proxy HTTP dan HTTP didukung oleh API Gateway. Untuk detailnya lihat [Otoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP di API Gateway](#).

Topik

- [Siapkan permintaan metode di API Gateway](#)
- [Siapkan respons metode di API Gateway](#)
- [Siapkan metode menggunakan konsol API Gateway](#)

Siapkan permintaan metode di API Gateway

Menyiapkan permintaan metode melibatkan melakukan tugas-tugas berikut, setelah membuat [RestApi](#) sumber daya:

1. Membuat API baru atau memilih entitas [Sumber Daya](#) API yang ada.

2. Membuat sumber daya [Metode](#) API yang merupakan kata kerja HTTP spesifik pada API Resource baru atau yang dipilih. Tugas ini dapat dibagi lagi menjadi sub tugas berikut:

- Menambahkan metode HTTP ke permintaan metode
- Mengkonfigurasi parameter permintaan
- Mendefinisikan model untuk badan permintaan
- Memberlakukan skema otorisasi
- Mengaktifkan validasi permintaan

Anda dapat melakukan tugas-tugas ini menggunakan metode berikut:

- [Konsol API Gateway](#)
- AWS CLI [perintah \(create-resource dan put-method\)](#)
- AWS [Fungsi SDK \(misalnya, di Node.js, createResource dan putMethod\)](#)
- [API Gateway REST API \(sumber daya: buat dan metode:put\).](#)

Topik

- [Siapkan sumber daya API](#)
- [Siapkan metode HTTP](#)
- [Siapkan parameter permintaan metode](#)
- [Siapkan model permintaan metode](#)
- [Siapkan otorisasi permintaan metode](#)
- [Siapkan validasi permintaan metode](#)

Siapkan sumber daya API

Di API Gateway API, Anda mengekspos sumber daya yang dapat dialamatkan sebagai pohon entitas [Sumber Daya](#) API, dengan sumber daya root (/) di bagian atas hierarki. Sumber daya root relatif terhadap URL dasar API, yang terdiri dari titik akhir API dan nama panggung. Di konsol API Gateway, URI dasar ini disebut sebagai URI Invoke dan ditampilkan di editor tahap API setelah API diterapkan.

Titik akhir API dapat berupa nama host default atau nama domain khusus. Nama host default adalah dari format berikut:

```
{api-id}.execute-api.{region}.amazonaws.com
```

Dalam format ini, ini `{api-id}` mewakili pengenal API yang dihasilkan oleh API Gateway.

`{region}` Variabel mewakili AWS Region (misalnya, `us-east-1`) yang Anda pilih saat membuat API. Nama domain kustom adalah nama yang ramah pengguna di bawah domain internet yang valid. Misalnya, jika Anda telah mendaftarkan domain `internetexample.com`, salah satu dari `*.example.com` adalah nama domain kustom yang valid. Untuk informasi selengkapnya, lihat [membuat nama domain kustom](#).

Untuk [API PetStore sampel](#), sumber daya root (/) mengekspos toko hewan peliharaan. Sumber `/pets` mewakili koleksi hewan peliharaan yang tersedia di toko hewan peliharaan. Ini `/pets/{petId}` mengekspos hewan peliharaan individu dari pengenal yang diberikan (`petId`). Parameter jalur `{petId}` adalah bagian dari parameter permintaan.

Untuk menyiapkan sumber daya API, Anda memilih sumber daya yang ada sebagai induknya, lalu membuat sumber daya turunan di bawah sumber daya induk ini. Anda mulai dengan sumber daya root sebagai induk, menambahkan sumber daya ke induk ini, menambahkan sumber daya lain ke sumber daya turunan ini sebagai induk baru, dan seterusnya, ke pengenal induknya. Kemudian Anda menambahkan sumber daya bernama ke induk.

Perintah [get-resources](#) berikut mengambil semua sumber daya API:

```
aws apigateway get-resources --rest-api-id apiId
```

Untuk PetStore contoh API, outputnya terlihat seperti berikut:

```
{
  "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"
    }
]
```

Setiap item mencantumkan pengidentifikasi sumber daya (`id`) dan, kecuali sumber daya root, induk langsungnya (`parentId`), serta nama sumber daya (`pathPart`). Sumber daya root istimewa karena tidak memiliki induk. Setelah memilih sumber daya sebagai induk, gunakan perintah berikut untuk menambahkan sumber daya anak:

```
aws apigateway create-resource --rest-api-id apiId \
    --parent-id parentId \
    --path-part resourceName
```

Misalnya, untuk menambahkan makanan hewan untuk dijual di PetStore situs web, gunakan perintah berikut:

```
aws apigateway create-resource --rest-api-id a1b2c3 \
    --parent-id svzr2028x8 \
    --path-part food
```

Outputnya akan terlihat seperti berikut:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Menggunakan sumber daya proxy untuk merampingkan penyiapan API

Seiring pertumbuhan bisnis, PetStore pemilik dapat memutuskan untuk menambahkan makanan, mainan, dan barang-barang terkait hewan peliharaan lainnya untuk dijual. Untuk mendukung ini, Anda dapat menambahkan `/food`, `/toys`, dan sumber daya lainnya di bawah sumber daya root. Di bawah setiap kategori penjualan, Anda mungkin juga ingin menambahkan lebih banyak sumber

daya, seperti `/food/{type}/{item}`, `/toys/{type}/{item}`, dll. Ini bisa membosankan. Jika Anda memutuskan untuk menambahkan lapisan tengah `{subtype}` ke jalur sumber daya untuk mengubah hierarki jalur menjadi `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, dll., Perubahan akan merusak pengaturan API yang ada. Untuk menghindari hal ini, Anda dapat menggunakan [sumber daya proxy](#) API Gateway untuk mengekspos sekumpulan sumber daya API sekaligus.

API Gateway mendefinisikan sumber daya proxy sebagai placeholder untuk sumber daya yang akan ditentukan saat permintaan dikirimkan. Sumber daya proxy diekspresikan oleh parameter jalur khusus `{proxy+}`, sering disebut sebagai parameter jalur serakah. Tanda tersebut menunjukkan sumber daya anak mana pun yang ditambahkan padanya. `/parent/{proxy+}` Placeholder adalah singkatan dari sumber daya apa pun yang cocok dengan pola jalur. `/parent/*` Anda dapat menggunakan string apa pun untuk nama parameter jalur serakah.

Perintah [create-resource](#) berikut membuat sumber daya proxy di bawah root (`/`): `/ {proxy+}`

```
aws apigateway create-resource --rest-api-id apiId \
  --parent-id rootResourceId \
  --path-part {proxy+}
```

Outputnya akan terlihat seperti berikut:

```
{
  "path": "/{proxy+}",
  "pathPart": "{proxy+}",
  "id": "234jdr",
  "parentId": "svzr2028x8"
}
```

Untuk contoh PetStore API, Anda dapat menggunakan `/ {proxy+}` untuk mewakili `/pets` dan `/pets/{petId}`. Sumber daya proxy ini juga dapat mereferensikan sumber daya lain (yang ada atau to-be-added) `/food/{type}/{item}`, seperti `/toys/{type}/{item}`, dll., Atau `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, dll. Pengembang backend menentukan hierarki sumber daya dan pengembang klien bertanggung jawab untuk memahaminya. API Gateway hanya meneruskan apa pun yang dikirimkan klien ke backend.

API dapat memiliki lebih dari satu sumber daya proxy. Misalnya, sumber daya proxy berikut diizinkan dalam API, dengan asumsi `/parent/{proxy+}` bukan induk yang sama dengan `/parent/{child}/{proxy+}`

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Ketika sumber daya proxy memiliki saudara kandung non-proxy, sumber daya saudara dikecualikan dari representasi sumber daya proxy. Untuk contoh sebelumnya, `/ {proxy+}` mengacu pada sumber daya apa pun di bawah sumber daya root kecuali sumber daya. `/parent[/*]` Dengan kata lain, permintaan metode terhadap sumber daya tertentu lebih diutamakan daripada permintaan metode terhadap sumber daya generik pada tingkat hierarki sumber daya yang sama.

Tabel berikut menunjukkan cara API Gateway merutekan permintaan ke sumber daya berikut untuk prod tahap API.

```
ANY /{proxy+}
GET /pets/{proxy+}
GET /pets/dog
```

Permintaan	Rute yang dipilih	Penjelasan
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/prod/pets/dog	GET /pets/dog	Permintaan sepenuhnya cocok dengan sumber daya ini.
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/prod/pets/cats	GET /pets/{proxy+}	Variabel jalur <code>/pets/{proxy+}</code> serakah menangkap permintaan ini.
GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/prod/animals	GET /{proxy+}	Variabel jalur <code>/ {proxy+}</code> serakah menangkap permintaan ini.

Sumber daya proxy tidak dapat memiliki sumber daya anak. Sumber daya API apa pun `{proxy+}` setelahnya berlebihan dan ambigu. Sumber daya proxy berikut tidak diizinkan dalam API.

```
{proxy+}/child  
/parent/{proxy+}/{child}  
/parent/{child}/{proxy+}/{grandchild+}
```

Siapkan metode HTTP

[Permintaan metode API dienkapsulasi oleh sumber daya Metode API Gateway](#). Untuk mengatur permintaan metode, Anda harus terlebih dahulu membuat instance Method sumber daya, menyetel setidaknya metode HTTP dan jenis otorisasi pada metode.

Terkait erat dengan sumber daya proxy, API Gateway mendukung metode HTTPANY. ANYMetode ini mewakili setiap metode HTTP yang akan disediakan pada waktu berjalan. Ini memungkinkan Anda untuk menggunakan penyiapan metode API tunggal untuk semua metode HTTP yang didukungDELETE,GET,HEAD,OPTIONS,PATCH,POST, danPUT.

Anda dapat mengatur ANY metode pada sumber daya non-proxy juga. Menggabungkan ANY metode dengan sumber daya proxy, Anda mendapatkan penyiapan metode API tunggal untuk semua metode HTTP yang didukung terhadap sumber daya API apa pun. Selain itu, backend dapat berkembang tanpa merusak pengaturan API yang ada.

Sebelum menyiapkan metode API, pertimbangkan siapa yang dapat memanggil metode tersebut. Atur jenis otorisasi sesuai dengan rencana Anda. Untuk akses terbuka, atur keNONE. Untuk menggunakan izin IAM, atur jenis otorisasi ke. AWS_IAM Untuk menggunakan fungsi otorisasi Lambda, setel properti ini ke. CUSTOM Untuk menggunakan kumpulan pengguna Amazon Cognito, setel jenis otorisasi ke. COGNITO_USER_POOLS

Perintah [put-method](#) berikut membuat permintaan metode untuk ANY kata kerja menggunakan izin IAM untuk mengontrol aksesnya.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method ANY \  
  --authorization-type AWS_IAM
```

Untuk membuat permintaan metode API dengan jenis otorisasi yang berbeda, lihat[the section called “Siapkan otorisasi permintaan metode”](#).

Siapkan parameter permintaan metode

Parameter permintaan metode adalah cara bagi klien untuk menyediakan data input atau konteks eksekusi yang diperlukan untuk menyelesaikan permintaan metode. Parameter metode dapat berupa parameter jalur, header, atau parameter string kueri. Sebagai bagian dari pengaturan permintaan metode, Anda harus mendeklarasikan parameter permintaan yang diperlukan agar tersedia untuk klien. Untuk integrasi non-proxy, Anda dapat menerjemahkan parameter permintaan ini ke formulir yang kompatibel dengan persyaratan backend.

Misalnya, untuk permintaan GET `/pets/{petId}` metode, variabel `{petId}` jalur adalah parameter permintaan yang diperlukan. Anda dapat mendeklarasikan parameter jalur ini saat memanggil `put-method` perintah. AWS CLI Perintah [put-method](#) berikut menciptakan metode dengan parameter path yang diperlukan:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id rjkmth \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters method.request.path.petId=true
```

Jika parameter tidak diperlukan, Anda dapat mengaturnya ke `false` dalam `request-parameters`. Misalnya, jika GET `/pets` metode menggunakan parameter string kueri opsional `daritype`, dan parameter header opsional `age`, Anda dapat mendeklarasikannya menggunakan perintah [put-method](#) berikut:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters \  
    method.request.querystring.type=false,method.request.header.age=false
```

Alih-alih formulir singkat ini, Anda dapat menggunakan string JSON untuk mengatur nilai `request-parameters`:

```
'{"method.request.querystring.type":false,"method.request.header.age":false}'
```

Dengan pengaturan ini, klien dapat menanyakan hewan peliharaan berdasarkan jenis:

```
GET /pets?type=dog
```

Dan klien dapat menanyakan anjing-anjingnya yang masih kecil sebagai berikut:

```
GET /pets?type=dog  
age:puppy
```

Untuk informasi tentang cara memetakan parameter permintaan metode ke parameter permintaan integrasi, lihat [the section called “Integrasi”](#).

Siapkan model permintaan metode

Untuk metode API yang dapat mengambil data input dalam payload, Anda dapat menggunakan model. Sebuah model diekspresikan dalam [skema JSON draft 4](#) dan menjelaskan struktur data dari badan permintaan. Dengan model, klien dapat menentukan bagaimana membangun payload permintaan metode sebagai input. Lebih penting lagi, API Gateway menggunakan model untuk [memvalidasi permintaan](#), [menghasilkan SDK](#), dan menginisialisasi template pemetaan untuk menyiapkan integrasi di konsol API Gateway. Untuk informasi tentang cara membuat [model](#), lihat [Memahami model data](#).

Tergantung pada jenis konten, payload metode dapat memiliki format yang berbeda. Sebuah model diindeks terhadap jenis media dari muatan yang diterapkan. API Gateway menggunakan header Content-Type permintaan untuk menentukan jenis konten. Untuk mengatur model permintaan metode, tambahkan pasangan kunci-nilai "*media-type*" : "*model-name*" format ke requestModels peta saat memanggil perintah. AWS CLI put-method

Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan \$default sebagai kunci.

Misalnya, untuk menyetel model pada payload JSON dari permintaan POST /pets metode API PetStore contoh, Anda dapat menggunakan perintah [put-method](#) berikut:

```
aws apigateway put-method \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxz2j \  
  --http-method POST \  
  --authorization-type "NONE" \  
  --request-models '{"application/json":"petModel"}'
```


Di sini, `petModel` adalah nilai name properti dari [Model](#) sumber daya yang menggambarkan hewan peliharaan. Definisi skema yang sebenarnya dinyatakan sebagai nilai string JSON dari [schema](#) properti sumber daya. `Model`

Di Java, atau SDK lain yang diketik kuat, dari API, data input dilemparkan sebagai `petModel` kelas yang berasal dari definisi skema. Dengan model permintaan, data input dalam SDK yang dihasilkan dilemparkan ke `Empty` kelas, yang berasal dari `Empty` model default. Dalam hal ini, klien tidak dapat membuat instance kelas data yang benar untuk memberikan input yang diperlukan.

Siapkan otorisasi permintaan metode

Untuk mengontrol siapa yang dapat memanggil metode API, Anda dapat mengonfigurasi [jenis otorisasi](#) pada metode. Anda dapat menggunakan jenis ini untuk memberlakukan salah satu otorisasi yang didukung, termasuk peran dan kebijakan IAM (`AWS_IAM`), kumpulan pengguna Amazon Cognito (`COGNITO_USER_POOLS`), atau pemberi otorisasi Lambda (`CUSTOM`).

Untuk menggunakan izin IAM untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke `AWS_IAM`. Saat Anda menyetel opsi ini, API Gateway memverifikasi tanda tangan pemanggil berdasarkan kredensial pemanggil. Jika pengguna terverifikasi memiliki izin untuk memanggil metode, ia menerima permintaan tersebut. Jika tidak, ia menolak permintaan dan pemanggil menerima respons kesalahan yang tidak sah. Panggilan ke metode tidak berhasil kecuali pemanggil memiliki izin untuk memanggil metode API. Kebijakan IAM berikut memberikan izin kepada pemanggil untuk memanggil metode API apa pun yang dibuat dalam metode yang sama: Akun AWS

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": "arn:aws:execute-api:*:*:*"
    }
  ]
}
```

Untuk informasi selengkapnya, lihat [the section called “Gunakan izin IAM”](#).

Saat ini, Anda hanya dapat memberikan kebijakan ini kepada pengguna, grup, dan peran dalam pemilik API Akun AWS. Pengguna dari yang berbeda Akun AWS dapat memanggil metode API hanya jika diizinkan untuk mengambil peran dalam pemilik API Akun AWS dengan izin yang diperlukan untuk memanggil `execute-api:Invoke` tindakan. Untuk informasi tentang izin lintas akun, lihat [Menggunakan Peran IAM](#).

Anda dapat menggunakan AWS CLI, AWS SDK, atau klien REST API, seperti [Postman](#), yang mengimplementasikan [penandatanganan Signature Version 4 \(SigV4\)](#).

Untuk menggunakan otorisasi Lambda untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke `CUSTOM` dan setel properti `authorizer-id` input ke nilai properti dari otorisasi Lambda yang sudah ada. `id` Authorizer Lambda yang direferensikan dapat dari `TOKEN` tipe atau `REQUEST` Untuk informasi tentang membuat otorisasi Lambda, lihat. [the section called “Gunakan otorisasi Lambda”](#)

Untuk menggunakan kumpulan pengguna Amazon Cognito untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke `COGNITO_USER_POOLS` dan setel properti `authorizer-id` input ke nilai `id` properti `COGNITO_USER_POOLS` otorisasi yang telah dibuat. Untuk informasi tentang membuat otorisasi kumpulan pengguna Amazon Cognito, lihat. [the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”](#)

Siapkan validasi permintaan metode

Anda dapat mengaktifkan validasi permintaan saat menyiapkan permintaan metode API. Anda harus terlebih dahulu membuat [validator permintaan](#). [create-request-validator](#) Perintah berikut membuat validator permintaan `body-only`.

```
aws apigateway create-request-validator \  
  --rest-api-id 7zw9uyk9kl \  
  --name bodyOnlyValidator \  
  --validate-request-body \  
  --no-validate-request-parameters
```

Outputnya akan terlihat seperti berikut:

```
{  
  "validateRequestParameters": false,  
  "validateRequestBody": true,
```

```
"id": "jgppy6",  
"name": "bodyOnlyValidator"  
}
```

Anda dapat menggunakan validator permintaan ini, untuk menggunakan validasi permintaan sebagai bagian dari pengaturan permintaan metode. Perintah [put-method](#) berikut membuat permintaan metode yang mengharuskan badan permintaan masuk untuk mencocokkan PetModel dan memiliki dua parameter permintaan yang tidak diperlukan:

```
aws apigateway put-method \  
  --rest-api-id 7zw9uyk9kl \  
  --resource-id xdsvhp \  
  --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
```

Untuk menyertakan parameter permintaan dalam validasi permintaan, Anda harus mengatur `validateRequestParameters true` untuk validator permintaan, dan menetapkan parameter permintaan khusus ke `true` dalam perintah. `put-method`

Siapkan respons metode di API Gateway

Respons metode API merangkum output dari permintaan metode API yang akan diterima klien. Data output mencakup kode status HTTP, beberapa header, dan mungkin badan.

Dengan integrasi non-proxy, parameter dan badan respons yang ditentukan dapat dipetakan dari data respons integrasi terkait atau dapat diberikan nilai statis tertentu sesuai dengan pemetaan. Pemetaan ini ditentukan dalam respons integrasi. Pemetaan dapat menjadi transformasi identik yang melewati respons integrasi melalui apa adanya.

Dengan integrasi proxy, API Gateway meneruskan respons backend ke respons metode secara otomatis. Anda tidak perlu menyiapkan respons metode API. Namun, dengan integrasi proxy Lambda, fungsi Lambda harus mengembalikan hasil format [keluaran ini agar API](#) Gateway berhasil memetakan respons integrasi ke respons metode.

[Secara terprogram, pengaturan respons metode sama dengan membuat MethodResponsesumber daya API Gateway dan menyetel properti StatusCode, ResponseParameters, dan ResponseModels.](#)

Saat menyetel kode status untuk metode API, Anda harus memilih salah satu sebagai default untuk menangani respons integrasi kode status yang tidak terduga. Masuk akal untuk ditetapkan 500 sebagai default karena ini sama dengan mentransmisikan respons yang tidak dipetakan sebagai kesalahan sisi server. Untuk alasan instruksional, konsol API Gateway menetapkan 200 respons sebagai default. Tetapi Anda dapat mengatur ulang ke 500 respons.

Untuk mengatur respons metode, Anda harus membuat permintaan metode.

Mengatur kode status respons metode

Kode status dari respon metode mendefinisikan jenis respon. Misalnya, tanggapan 200, 400, dan 500 menunjukkan keberhasilan, kesalahan sisi klien dan respons kesalahan sisi server, masing-masing.

Untuk menyiapkan kode status respons metode, setel [statusCode](#) properti ke kode status HTTP. [put-method-response](#) Perintah berikut menciptakan respon 200 metode.

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxz2j \  
  --http-method GET \  
  --status-code 200
```

Siapkan parameter respons metode

Parameter respons metode menentukan header mana yang diterima klien sebagai respons terhadap permintaan metode terkait. Parameter respons juga menentukan target yang API Gateway memetakan parameter respons integrasi, sesuai dengan pemetaan yang ditentukan dalam respons integrasi metode API.

Untuk mengatur parameter respons metode, tambahkan ke [responseParameters](#) peta pasangan MethodResponse kunci-nilai format. "{parameter-name}": "{boolean}" [put-method-response](#) Perintah berikut menetapkan my-header header.

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxz2j \  
  --http-method GET \  
  --status-code 200 \  
  --response-parameters method.response.header.my-header=false
```

Siapkan model respons metode

Model respons metode mendefinisikan format badan respons metode. Menyiapkan model respons metode diperlukan saat Anda membuat SDK yang diketik kuat untuk API. Ini memastikan bahwa output dilemparkan ke kelas yang sesuai di Java atau Objective-C. Dalam kasus lain, pengaturan model adalah opsional.

Sebelum menyiapkan model respons, Anda harus terlebih dahulu membuat model di API Gateway. Untuk melakukannya, Anda dapat memanggil [create-model](#) perintah. Perintah [create-model](#) berikut membuat PetStorePet model untuk menggambarkan tubuh respons terhadap permintaan metode. GET /pets/{petId}

```
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" } \  
    } \  
  }'
```

Hasilnya dibuat sebagai [Model](#) sumber daya API Gateway.

Untuk mengatur model respons metode untuk menentukan format payload, tambahkan pasangan nilai kunci “application/json”:”PetStorePet” ke peta sumber daya. [requestModelsMethodResponse](#) [put-method-response](#) Perintah berikut membuat respons metode yang menggunakan model respons untuk menentukan format payload:

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxz2j \  
  --http-method GET \  
  --status-code 200 \  
  --response-parameters method.response.header.my-header=false \  
  --model-id PetStorePet
```

```
--response-models '{"application/json":"PetStorePet"}'
```

Siapkan metode menggunakan konsol API Gateway

Saat membuat metode menggunakan konsol REST API, Anda mengonfigurasi permintaan integrasi dan permintaan metode. Secara default, API Gateway membuat respons 200 metode untuk metode Anda.

Petunjuk berikut menunjukkan cara mengedit pengaturan permintaan metode dan cara membuat respons metode tambahan untuk metode Anda.

Topik

- [Mengedit permintaan metode API Gateway di konsol API Gateway](#)
- [Menyiapkan respons metode API Gateway menggunakan konsol API Gateway](#)

Mengedit permintaan metode API Gateway di konsol API Gateway

Instruksi ini mengasumsikan Anda telah membuat permintaan metode Anda. Untuk informasi selengkapnya tentang cara membuat metode, lihat [the section called “ Siapkan permintaan integrasi menggunakan konsol”](#).

1. Di panel Resources, pilih metode Anda, lalu pilih tab Permintaan metode.
2. Di bagian Pengaturan permintaan metode, pilih Edit.
3. Untuk Otorisasi, pilih otorisasi yang tersedia.
 - a. Untuk mengaktifkan akses terbuka ke metode untuk pengguna mana pun, pilih Tidak Ada. Langkah ini dapat dilewati jika pengaturan default belum diubah.
 - b. Untuk menggunakan izin IAM untuk mengontrol akses klien ke metode, pilih. AWS_IAM Dengan pilihan ini, hanya pengguna peran IAM dengan kebijakan IAM yang benar dilampirkan yang diizinkan untuk memanggil metode ini.

Untuk membuat peran IAM, tentukan kebijakan akses dengan format seperti berikut:

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "*"  
      },  
      "Action": "execute-api:Invoke",  
      "Resource": "*"  
    }  
  ]  
}
```

```
{
  "Effect": "Allow",
  "Action": [
    "execute-api:Invoke"
  ],
  "Resource": [
    "arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/"
  ]
}
```

Dalam kebijakan akses ini, `arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/` adalah ARN metode Anda. Anda dapat menemukan ARN metode Anda dengan memilih metode pada halaman Sumber Daya. Untuk informasi selengkapnya tentang menyetel izin IAM, lihat. [Kontrol akses ke REST API dengan izin IAM](#)

Untuk membuat peran IAM, Anda dapat menyesuaikan instruksi dalam tutorial berikut, [???](#).

- c. Untuk menggunakan otorisasi Lambda, pilih token atau otorisasi permintaan. Buat otorisasi Lambda agar pilihan ini ditampilkan di menu tarik-turun. Untuk informasi tentang cara membuat otorisasi Lambda, lihat. [Gunakan otorisasi API Gateway Lambda](#)
 - d. Untuk menggunakan kumpulan pengguna Amazon Cognito, pilih kumpulan pengguna yang tersedia di bawah otorisasi kumpulan pengguna Cognito. Buat kumpulan pengguna di Amazon Cognito dan otorisasi kumpulan pengguna Amazon Cognito di API Gateway agar pilihan ini ditampilkan di menu tarik-turun. Untuk informasi tentang cara membuat otorisasi kumpulan pengguna Amazon Cognito, lihat. [Kontrol akses ke REST APIs menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi](#)
4. Untuk menentukan validasi permintaan, pilih nilai dari menu tarik-turun Permintaan Validator. Untuk menonaktifkan validasi permintaan, pilih Tidak Ada. Untuk informasi selengkapnya tentang setiap opsi, lihat [Minta validasi untuk REST APIs di API Gateway](#).
 5. Pilih kunci API yang diperlukan untuk meminta kunci API. Saat diaktifkan, kunci API digunakan dalam [rencana penggunaan](#) untuk membatasi lalu lintas klien.
 6. (Opsional) Untuk menetapkan nama operasi di Java SDK API ini, yang dihasilkan oleh API Gateway, untuk nama Operasi, masukkan nama. Misalnya, untuk permintaan metode GET / pets/{petId}, nama operasi Java SDK yang sesuai adalah, secara default, GetPetsPetId. Nama ini dibangun dari kata kerja HTTP metode (GET) dan nama variabel jalur sumber daya

(PetsdanPetId). Jika Anda menetapkan nama operasi sebagaigetPetById, nama operasi SDK menjadiGetPetById.

7. Untuk menambahkan parameter string kueri ke metode, lakukan hal berikut:
 - a. Pilih parameter string Kueri URL, lalu pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan nama parameter string kueri.
 - c. Pilih Diperlukan jika parameter string kueri yang baru dibuat akan digunakan untuk validasi permintaan. Untuk informasi selengkapnya tentang validasi permintaan, lihat[Minta validasi untuk REST APIs di API Gateway](#).
 - d. Pilih Caching jika parameter string kueri yang baru dibuat akan digunakan sebagai bagian dari kunci caching. Untuk informasi lebih lanjut tentang caching, lihat[Gunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache](#).

Untuk menghapus parameter string kueri, pilih Hapus.

8. Untuk menambahkan parameter header ke metode, lakukan hal berikut:
 - a. Pilih header permintaan HTTP, lalu pilih Tambah header.
 - b. Untuk Nama, masukkan nama header.
 - c. Pilih Diperlukan jika header yang baru dibuat akan digunakan untuk validasi permintaan. Untuk informasi selengkapnya tentang validasi permintaan, lihat[Minta validasi untuk REST APIs di API Gateway](#).
 - d. Pilih Caching jika header yang baru dibuat akan digunakan sebagai bagian dari kunci caching. Untuk informasi lebih lanjut tentang caching, lihat[Gunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache](#).

Untuk menghapus header, pilih Hapus.

9. Untuk mendeklarasikan format payload permintaan metode dengan,, atau kata kerja PATCH HTTP POSTPUT, pilih Request body, dan lakukan hal berikut:
 - a. Pilih Tambah model.
 - b. Untuk Content-type, masukkan tipe MIME (misalnya,. application/json
 - c. Untuk Model, pilih model dari menu tarik-turun. Model yang tersedia saat ini untuk API mencakup default Empty dan Error model serta model apa pun yang telah Anda buat dan tambahkan ke koleksi [Model](#) API. Untuk informasi selengkapnya tentang membuat model, lihat[Model data untuk REST APIs](#).

 Note

Model ini berguna untuk memberi tahu klien tentang format data yang diharapkan dari muatan. Sangat membantu untuk menghasilkan template pemetaan kerangka. Penting untuk menghasilkan SDK API yang diketik dengan kuat dalam bahasa seperti Java, C #, Objective-C, dan Swift. Ini hanya diperlukan jika validasi permintaan diaktifkan terhadap muatan.

10. Pilih Simpan.

Menyiapkan respons metode API Gateway menggunakan konsol API Gateway

Metode API dapat memiliki satu atau lebih tanggapan. Setiap respons diindeks oleh kode status HTTP-nya. Secara default, konsol API Gateway menambahkan 200 respons ke respons metode. Anda dapat memodifikasinya, misalnya, agar metode dikembalikan 201 sebagai gantinya. Anda dapat menambahkan tanggapan lain, misalnya, untuk penolakan akses dan 409 500 untuk variabel tahap yang tidak diinisialisasi yang digunakan.

Untuk menggunakan konsol API Gateway untuk memodifikasi, menghapus, atau menambahkan respons ke metode API, ikuti petunjuk berikut.

1. Di panel Resources, pilih metode Anda, lalu pilih tab Respons metode. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bagian Pengaturan respons metode, pilih Buat respons.
3. Untuk kode status HTTP, masukkan kode status HTTP seperti 200, 400, atau 500.

Ketika respons yang dikembalikan ke backend tidak memiliki respons metode yang sesuai yang ditentukan, API Gateway gagal mengembalikan respons ke klien. Sebaliknya, ia mengembalikan respons 500 Internal server error kesalahan.

4. Pilih Tambahkan header.
5. Untuk nama Header, masukkan nama.

Untuk mengembalikan header dari backend ke klien, tambahkan header dalam respons metode.

6. Pilih Tambahkan model untuk menentukan format badan respons metode.

Masukkan jenis media payload respons untuk jenis Konten dan pilih model dari menu tarik-turun Model.

7. Pilih Simpan.

Untuk mengubah respons yang ada, navigasikan ke respons metode Anda, lalu pilih Edit. Untuk mengubah kode status HTTP, pilih Hapus dan buat respons metode baru.

Untuk setiap respons yang dikembalikan dari backend, Anda harus memiliki respons yang kompatibel yang dikonfigurasi sebagai respons metode. Namun, header respons metode konfigurasi dan model payload bersifat opsional kecuali Anda memetakan hasil dari backend ke respons metode sebelum kembali ke klien. Selain itu, model payload respons metode penting jika Anda membuat SDK yang diketik kuat untuk API Anda.

Kontrol dan kelola akses ke REST APIs di API Gateway

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke API Anda.

Anda dapat menggunakan mekanisme berikut untuk otentikasi dan otorisasi:

- Kebijakan sumber daya memungkinkan Anda membuat kebijakan berbasis sumber daya untuk mengizinkan atau menolak akses ke APIs dan metode Anda dari alamat IP sumber tertentu atau titik akhir VPC. Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan sumber daya API Gateway”](#).
- Peran dan kebijakan AWS IAM standar menawarkan kontrol akses yang fleksibel dan kuat yang dapat diterapkan ke seluruh API atau metode individual. Peran dan kebijakan IAM dapat digunakan untuk mengontrol siapa yang dapat membuat dan mengelola Anda APIs, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat [the section called “Gunakan izin IAM”](#).
- Tag IAM dapat digunakan bersama dengan kebijakan IAM untuk mengontrol akses. Untuk informasi selengkapnya, lihat [the section called “Kontrol Akses Berbasis Atribut”](#).
- Kebijakan titik akhir untuk titik akhir VPC antarmuka [memungkinkan Anda melampirkan kebijakan sumber daya IAM ke titik akhir VPC antarmuka untuk meningkatkan keamanan pribadi Anda. APIs](#) Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#).
- Pengotorisasi Lambda adalah fungsi Lambda yang mengontrol akses ke metode REST API menggunakan otentikasi token pembawa — serta informasi yang dijelaskan oleh header, jalur, string kueri, variabel tahap, atau parameter permintaan variabel konteks. Authorizer Lambda digunakan untuk mengontrol siapa yang dapat menjalankan metode REST API. Untuk informasi selengkapnya, lihat [the section called “Gunakan otorisasi Lambda”](#).

- Kumpulan pengguna Amazon Cognito memungkinkan Anda membuat solusi otentikasi dan otorisasi yang dapat disesuaikan untuk REST Anda. APIs Kumpulan pengguna Amazon Cognito digunakan untuk mengontrol siapa yang dapat menjalankan metode REST API. Untuk informasi selengkapnya, lihat [the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”](#).

Anda dapat menggunakan mekanisme berikut untuk melakukan tugas lain yang terkait dengan kontrol akses:

- Berbagi sumber daya lintas asal (CORS) memungkinkan Anda mengontrol cara REST API merespons permintaan sumber daya lintas domain. Untuk informasi selengkapnya, lihat [the section called “CORS”](#).
- Sertifikat SSL sisi klien dapat digunakan untuk memverifikasi bahwa permintaan HTTP ke sistem backend Anda berasal dari API Gateway. Untuk informasi selengkapnya, lihat [the section called “Sertifikat Klien”](#).
- AWS WAF dapat digunakan untuk melindungi API Gateway API Anda dari eksploitasi web umum. Untuk informasi selengkapnya, lihat [the section called “AWS WAF”](#).

Anda dapat menggunakan mekanisme berikut untuk melacak dan membatasi akses yang telah Anda berikan kepada klien yang berwenang:

- Paket penggunaan memungkinkan Anda memberikan kunci API kepada pelanggan Anda—lalu melacak dan membatasi penggunaan tahapan dan metode API Anda untuk setiap kunci API. Lihat informasi yang lebih lengkap di [the section called “Paket penggunaan”](#).

Kontrol akses ke REST API dengan kebijakan sumber daya API Gateway

Kebijakan sumber daya Amazon API Gateway adalah dokumen kebijakan JSON yang Anda lampirkan ke API untuk mengontrol apakah prinsipal tertentu (biasanya peran atau grup IAM) dapat memanggil API. Anda dapat menggunakan kebijakan sumber daya API Gateway untuk memungkinkan API Anda dipanggil dengan aman oleh:

- Pengguna dari AWS akun tertentu.
- Rentang alamat IP sumber tertentu atau blok CIDR.
- Cloud pribadi virtual tertentu (VPCs) atau titik akhir VPC (di akun apa pun).

Anda dapat melampirkan kebijakan sumber daya untuk semua jenis titik akhir API di API Gateway dengan menggunakan, AWS CLI Konsol Manajemen AWS, atau. AWS SDKs Untuk [pribadi APIs](#), Anda dapat menggunakan kebijakan sumber daya bersama dengan kebijakan titik akhir VPC untuk mengontrol prinsipal mana yang memiliki akses ke sumber daya dan tindakan mana. Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#).

Kebijakan sumber daya API Gateway berbeda dari kebijakan berbasis identitas IAM. Kebijakan berbasis identitas IAM dilampirkan pada pengguna, grup, atau peran IAM dan menentukan tindakan apa yang dapat dilakukan identitas tersebut pada sumber daya mana. Kebijakan sumber daya API Gateway dilampirkan ke sumber daya. Anda dapat menggunakan kebijakan sumber daya API Gateway bersama dengan kebijakan IAM. Untuk informasi lebih lanjut, lihat [Kebijakan Berbasis Identitas dan Kebijakan Berbasis Sumber Daya](#).

Topik

- [Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway](#)
- [Bagaimana kebijakan sumber daya API Gateway memengaruhi alur kerja otorisasi](#)
- [Contoh kebijakan sumber daya API Gateway](#)
- [Membuat dan melampirkan kebijakan sumber daya API Gateway ke API](#)
- [AWS kunci kondisi yang dapat digunakan dalam kebijakan sumber daya API Gateway](#)

Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway

Halaman ini menjelaskan elemen dasar yang digunakan dalam kebijakan sumber daya Amazon API Gateway.

Kebijakan sumber daya ditentukan menggunakan sintaks yang sama dengan kebijakan IAM. Untuk informasi bahasa kebijakan selengkapnya, lihat [Gambaran Umum Kebijakan IAM](#) dan [Referensi AWS Identity and Access Management Kebijakan](#) di Panduan Pengguna IAM.

Untuk informasi tentang cara AWS layanan memutuskan apakah permintaan yang diberikan harus diizinkan atau ditolak, lihat [Menentukan Apakah Permintaan Diizinkan atau Ditolak](#).

Elemen umum dalam kebijakan akses

Dalam arti yang paling mendasar, kebijakan sumber daya berisi elemen-elemen berikut:

- Sumber daya - APIs adalah sumber daya Amazon API Gateway yang dapat Anda izinkan atau tolak izinnya. Dalam sebuah kebijakan, Anda menggunakan Nama Sumber Daya Amazon (ARN)

untuk mengidentifikasi sumber daya. Anda juga dapat menggunakan sintaks singkat, yang API Gateway secara otomatis diperluas ke ARN penuh saat Anda menyimpan kebijakan sumber daya. Untuk mempelajari selengkapnya, lihat [Contoh kebijakan sumber daya API Gateway](#).

Untuk format Resource elemen lengkap, lihat [Format sumber daya izin untuk menjalankan API di API Gateway](#).

- Tindakan - Untuk setiap sumber daya, Amazon API Gateway mendukung serangkaian operasi. Anda mengidentifikasi operasi sumber daya yang Anda izinkan (atau tolak) dengan menggunakan kata kunci tindakan.

Misalnya, `execute-api:Invoke` izin akan memungkinkan izin pengguna untuk memanggil API atas permintaan klien.

Untuk format Action elemen, lihat [Format tindakan izin untuk menjalankan API di API Gateway](#).

- Efek — Apa efeknya ketika pengguna meminta tindakan tertentu—ini bisa berupa salah satu atau. Allow Deny Anda juga dapat secara eksplisit menolak akses ke sumber daya, yang mungkin Anda lakukan untuk memastikan bahwa pengguna tidak dapat mengaksesnya, bahkan jika kebijakan lain memberikan akses.

Note

“Penyangkalan implisit” adalah hal yang sama dengan “tolak secara default”.

“Penyangkalan implisit” berbeda dari “penolakan eksplisit”. Untuk informasi selengkapnya, lihat [Perbedaan Antara Menyangkal Secara Default dan Penolakan Eksplisit](#).

- Principal — Akun atau pengguna mengizinkan akses ke tindakan dan sumber daya dalam pernyataan. Dalam kebijakan sumber daya, prinsipal adalah pengguna atau akun yang menerima izin ini.

Contoh kebijakan sumber daya berikut menunjukkan elemen kebijakan umum sebelumnya.

Kebijakan memberikan akses ke API di bawah yang ditentukan *account-id* dalam yang ditentukan *region* untuk setiap pengguna yang alamat IP sumbernya ada di blok *123.4.5.6/24* alamat. Kebijakan menolak semua akses ke API jika IP sumber pengguna tidak berada dalam jangkauan.

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"
      }
    }
  }
]
```

Bagaimana kebijakan sumber daya API Gateway memengaruhi alur kerja otorisasi

Saat API Gateway mengevaluasi kebijakan sumber daya yang dilampirkan ke API Anda, hasilnya dipengaruhi oleh jenis autentikasi yang telah Anda tetapkan untuk API, seperti yang diilustrasikan dalam diagram alur di bagian berikut.

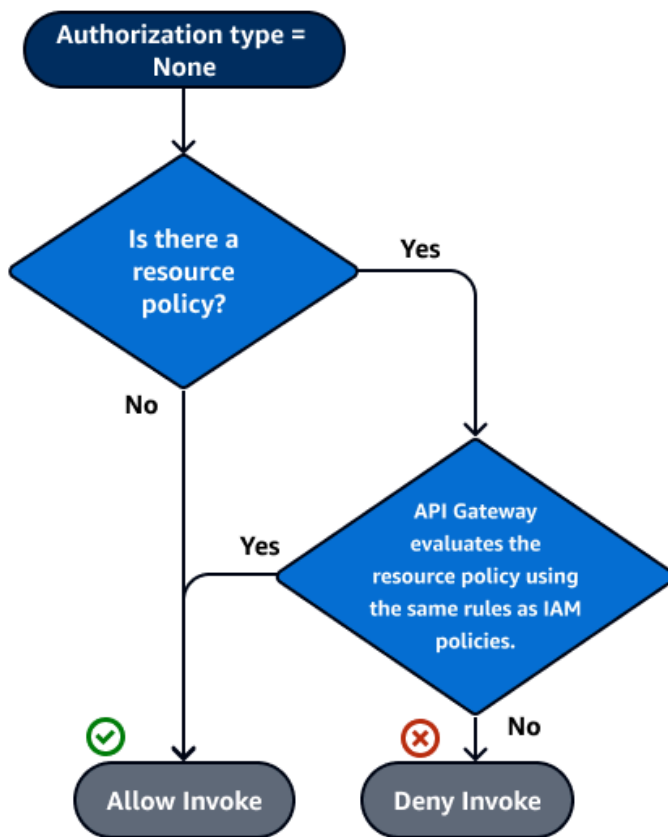
Topik

- [Kebijakan sumber daya API Gateway saja](#)
- [Lambda otorisasi dan kebijakan sumber daya](#)
- [Autentikasi IAM dan kebijakan sumber daya](#)
- [Autentikasi Amazon Cognito dan kebijakan sumber daya](#)
- [Tabel hasil evaluasi kebijakan](#)

Kebijakan sumber daya API Gateway saja

Dalam alur kerja ini, kebijakan sumber daya API Gateway dilampirkan ke API, tetapi tidak ada jenis otentikasi yang ditentukan untuk API. Evaluasi kebijakan melibatkan pencarian izin eksplisit

berdasarkan kriteria masuk penelepon. Penolakan implisit atau penolakan eksplisit apa pun menghasilkan penolakan penelepon.



Berikut ini adalah contoh kebijakan sumber daya semacam itu.

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"]
        }
      }
    }
  ]
}

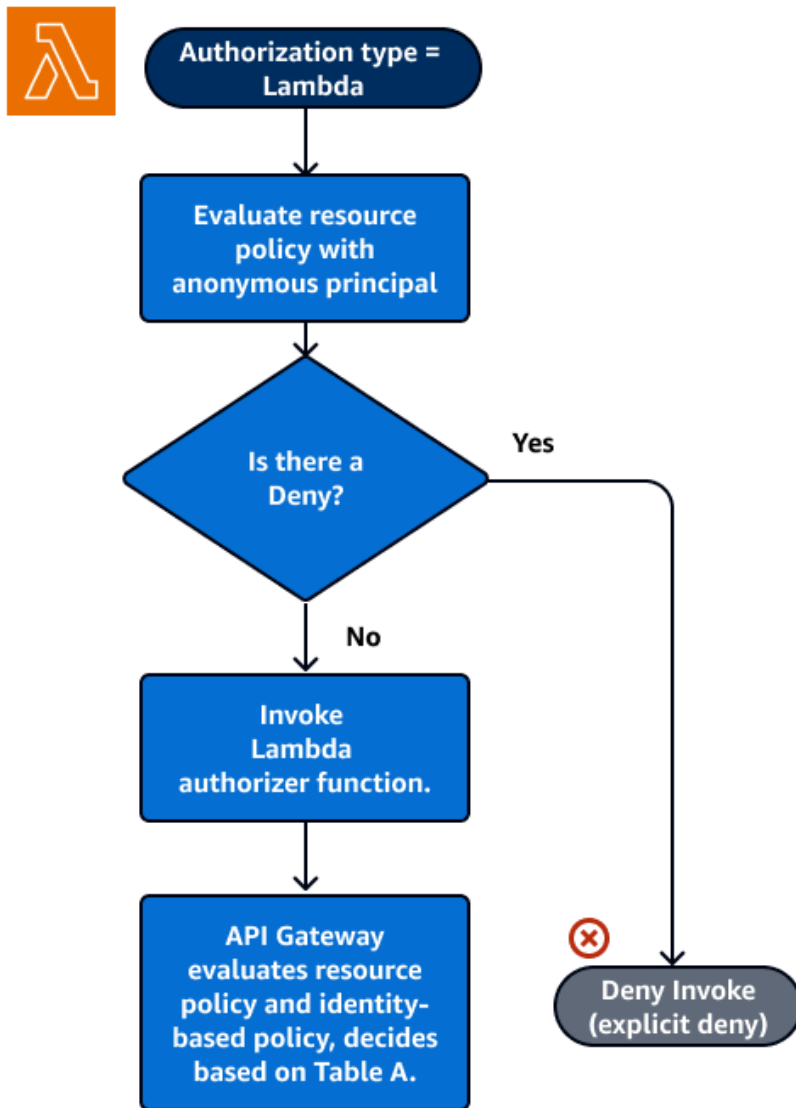
```

```
}  
  ]  
}
```

Lambda otorisasi dan kebijakan sumber daya

Dalam alur kerja ini, otorisasi Lambda dikonfigurasi untuk API selain kebijakan sumber daya. Kebijakan sumber daya dievaluasi dalam dua fase. Sebelum memanggil otorisasi Lambda, API Gateway terlebih dahulu mengevaluasi kebijakan dan memeriksa setiap penolakan eksplisit. Jika ditemukan, penelepon ditolak aksesnya segera. Jika tidak, otorisasi Lambda dipanggil, dan mengembalikan [dokumen kebijakan](#), yang dievaluasi bersama dengan kebijakan sumber daya. Jika otorisasi Anda menggunakan caching, API Gateway mungkin menampilkan dokumen kebijakan yang di-cache. Hasilnya ditentukan berdasarkan [Tabel A](#).

Contoh kebijakan sumber daya berikut mengizinkan panggilan hanya dari titik akhir VPC yang ID titik akhir VPC-nya. *vpce-1a2b3c4d* Selama evaluasi “pra-auth”, hanya panggilan yang berasal dari titik akhir VPC yang ditunjukkan dalam contoh yang diizinkan untuk bergerak maju dan mengevaluasi otorisasi Lambda. Semua panggilan yang tersisa diblokir. Alur kerja otorisasi ini sama jika Anda menggunakan nama domain khusus untuk API pribadi.



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": {
```

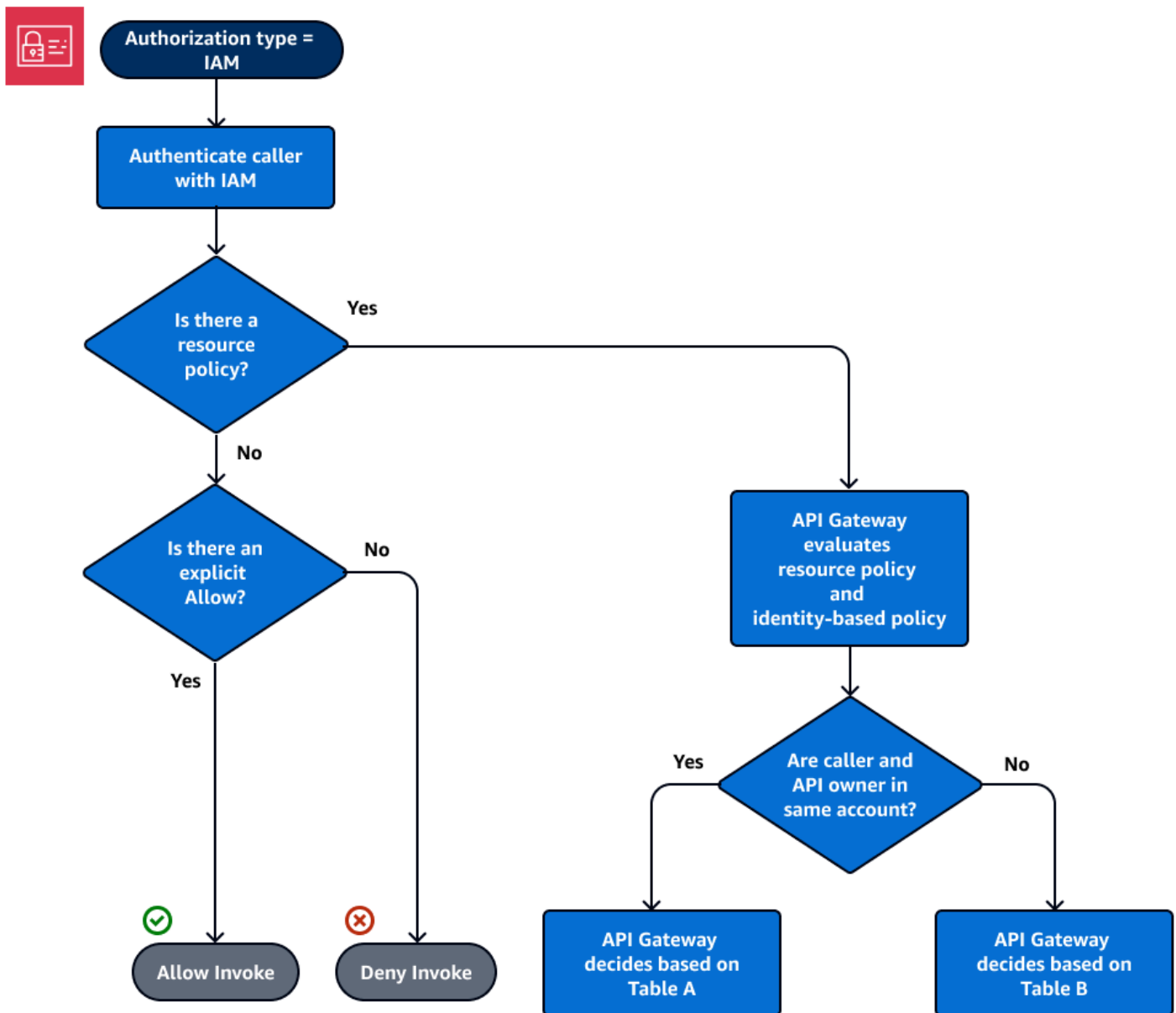
```
    "StringNotEquals": {  
      "aws:SourceVpce": "vpce-1a2b3c4d"  
    }  
  }  
]  
}
```

Autentikasi IAM dan kebijakan sumber daya

Dalam alur kerja ini, Anda mengonfigurasi autentikasi IAM untuk API selain kebijakan sumber daya. Setelah Anda mengautentikasi pengguna dengan layanan IAM, API mengevaluasi kebijakan yang dilampirkan pada pengguna dan kebijakan sumber daya. Hasilnya bervariasi berdasarkan apakah pemanggil berada di tempat yang sama Akun AWS atau terpisah Akun AWS, dari pemilik API.

Jika pemanggil dan pemilik API berasal dari akun terpisah, kebijakan IAM dan kebijakan sumber daya secara eksplisit mengizinkan pemanggil untuk melanjutkan. Untuk informasi lebih lanjut, lihat [Tabel B](#).

Namun, jika pemanggil dan pemilik API sama Akun AWS, maka kebijakan pengguna IAM atau kebijakan sumber daya harus secara eksplisit mengizinkan pemanggil untuk melanjutkan. Untuk informasi lebih lanjut, lihat [Tabel A](#).



Berikut ini adalah contoh kebijakan sumber daya lintas akun. Dengan asumsi kebijakan IAM berisi efek allow, kebijakan sumber daya ini hanya mengizinkan panggilan dari VPC yang ID VPCnya.

vpc-2f09a348 Untuk informasi lebih lanjut, lihat [Tabel B](#).

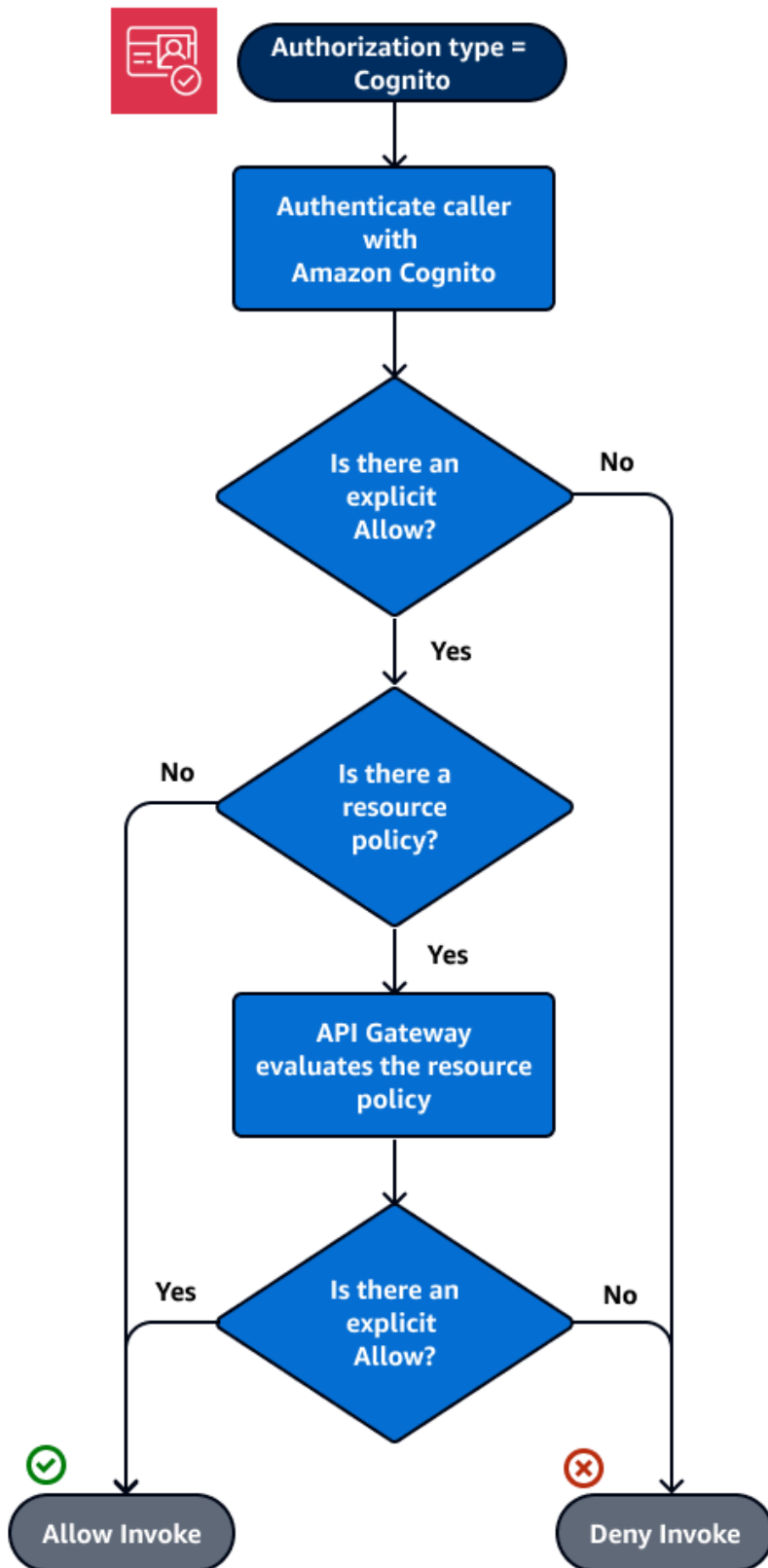
JSON

```
{
  "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"
        }
    }
}
```

Autentikasi Amazon Cognito dan kebijakan sumber daya

Dalam alur kerja ini, kumpulan [pengguna Amazon Cognito](#) dikonfigurasi untuk API selain kebijakan sumber daya. API Gateway pertama kali mencoba untuk mengautentikasi pemanggil melalui Amazon Cognito. Ini biasanya dilakukan melalui [token JWT](#) yang disediakan oleh penelepon. Jika otentikasi berhasil, kebijakan sumber daya dievaluasi secara independen, dan izin eksplisit diperlukan. Penyangkalan atau “tidak mengizinkan atau menyangkal” menghasilkan penyangkalan. Berikut ini adalah contoh kebijakan sumber daya yang dapat digunakan bersama dengan kumpulan pengguna Amazon Cognito.



Berikut ini adalah contoh kebijakan sumber daya yang mengizinkan panggilan hanya dari sumber tertentu IPs, dengan asumsi bahwa token otentikasi Amazon Cognito berisi izin. Untuk informasi lebih lanjut, lihat [Tabel 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"]
        }
      }
    }
  ]
}
```

Tabel hasil evaluasi kebijakan

Tabel A mencantumkan perilaku yang dihasilkan saat akses ke API Gateway API dikendalikan oleh kebijakan IAM atau otorisasi Lambda dan kebijakan sumber daya API Gateway, keduanya sama. Akun AWS

Kebijakan IAM (atau otorisasi Lambda)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Izinkan	Izinkan	Izinkan
Izinkan	Tidak Memungkinkan atau Menyangkal	Izinkan
Izinkan	Menyangkal	Penolakan jelas

Kebijakan IAM (atau otorisasi Lambda)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Tidak Memungkinkan atau Menyangkal	Izinkan	Izinkan
Tidak Memungkinkan atau Menyangkal	Tidak Memungkinkan atau Menyangkal	Penyangkalan Implisit
Tidak Memungkinkan atau Menyangkal	Menyangkal	Penolakan jelas
Menyangkal	Izinkan	Penolakan jelas
Menyangkal	Tidak Memungkinkan atau Menyangkal	Penolakan jelas
Menyangkal	Menyangkal	Penolakan jelas

Tabel B mencantumkan perilaku yang dihasilkan saat akses ke API Gateway API dikendalikan oleh kebijakan IAM atau otorisasi kumpulan pengguna Amazon Cognito dan kebijakan sumber daya API Gateway, yang berbeda. Akun AWS Jika salah satu diam (tidak mengizinkan atau menolak), akses lintas akun ditolak. Ini karena akses lintas akun mengharuskan kebijakan sumber daya dan kebijakan IAM atau otorisasi kumpulan pengguna Amazon Cognito secara eksplisit memberikan akses.

Kebijakan IAM (atau otorisasi kumpulan pengguna Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Izinkan	Izinkan	Izinkan
Izinkan	Tidak Memungkinkan atau Menyangkal	Penyangkalan Implisit
Izinkan	Menyangkal	Penolakan jelas
Tidak Memungkinkan atau Menyangkal	Izinkan	Penyangkalan Implisit

Kebijakan IAM (atau otorisasi kumpulan pengguna Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Tidak Memungkinkan atau Menyangkal	Tidak Memungkinkan atau Menyangkal	Penyangkalan Implisit
Tidak Memungkinkan atau Menyangkal	Menyangkal	Penolakan jelas
Menyangkal	Izinkan	Penolakan jelas
Menyangkal	Tidak Memungkinkan atau Menyangkal	Penolakan jelas
Menyangkal	Menyangkal	Penolakan jelas

Contoh kebijakan sumber daya API Gateway

Halaman ini menyajikan beberapa contoh kasus penggunaan umum untuk kebijakan sumber daya API Gateway.

Contoh kebijakan berikut menggunakan sintaks yang disederhanakan untuk menentukan sumber daya API. Sintaks yang disederhanakan ini adalah cara singkat yang dapat Anda rujuk ke sumber daya API, alih-alih menentukan Nama Sumber Daya Amazon (ARN) lengkap. API Gateway mengonversi sintaks yang disingkat menjadi ARN lengkap saat Anda menyimpan kebijakan. Misalnya, Anda dapat menentukan sumber daya `execute-api:/*stage-name/GET/pets` dalam kebijakan sumber daya. API Gateway mengonversi sumber daya menjadi `arn:aws:execute-api:us-east-2:123456789012:aabbccdde/stage-name/GET/pets` saat Anda menyimpan kebijakan sumber daya. API Gateway membangun ARN lengkap dengan menggunakan Wilayah saat ini, ID akun AWS Anda, dan ID REST API yang terkait dengan kebijakan sumber daya. Anda dapat menggunakan `execute-api:/*` untuk mewakili semua tahapan, metode, dan jalur di API saat ini. Untuk informasi tentang bahasa kebijakan akses, lihat [Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway](#).

Topik

- [Contoh: Izinkan peran di AWS akun lain untuk menggunakan API](#)
- [Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber](#)

- [Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber saat menggunakan API pribadi](#)
- [Contoh: Izinkan lalu lintas API pribadi berdasarkan titik akhir VPC atau VPC sumber](#)

Contoh: Izinkan peran di AWS akun lain untuk menggunakan API

Contoh kebijakan sumber daya berikut memberikan akses API dalam satu AWS akun ke dua peran di akun yang berbeda AWS melalui protokol [Signature Version 4 \(SigV4\)](#) atau Signature [Version 4a \(Sigv4a\)](#). Secara khusus, peran pengembang dan administrator untuk AWS akun yang diidentifikasi oleh *account-id-2* diberikan `execute-api:Invoke` tindakan untuk menjalankan GET tindakan pada `/pets` sumber daya (API) di AWS akun Anda.

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"
      ]
    }
  ]
}
```

Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber

Contoh kebijakan sumber daya berikut menyangkal (memblokir) lalu lintas masuk ke API dari dua blok alamat IP sumber yang ditentukan.

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" ]
        }
      }
    }
  ]
}
```

Jika Anda menggunakan kebijakan pengguna IAM atau kebijakan sumber daya API Gateway untuk mengontrol akses ke API Gateway atau API Gateway apa pun APIs, konfirmasi bahwa kebijakan Anda diperbarui untuk menyertakan rentang IPv6 alamat. Kebijakan yang tidak diperbarui untuk menangani IPv6 alamat dapat memengaruhi akses klien ke API Gateway saat mereka mulai menggunakan titik akhir dualstack. Untuk informasi selengkapnya, lihat [the section called “Menggunakan IPv6 alamat dalam kebijakan IAM”](#).

Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber saat menggunakan API pribadi

Contoh kebijakan sumber daya berikut menyangkal (memblokir) lalu lintas masuk ke API pribadi dari dua blok alamat IP sumber yang ditentukan. Saat menggunakan pribadi APIs, titik akhir VPC untuk `execute-api` menulis ulang alamat IP sumber asli. `aws:VpcSourceIpKondisi` memfilter permintaan terhadap alamat IP pemohon asli.

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"]
        }
      }
    }
  ]
}
```

Contoh: Izinkan lalu lintas API pribadi berdasarkan titik akhir VPC atau VPC sumber

Contoh kebijakan sumber daya berikut mengizinkan lalu lintas masuk ke API pribadi hanya dari virtual private cloud (VPC) atau titik akhir VPC tertentu.

Contoh kebijakan sumber daya ini menentukan VPC sumber:

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"
        }
      }
    }
  ]
}
```

Contoh kebijakan sumber daya ini menentukan titik akhir VPC sumber:

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"
            }
        }
    }
]
}

```

Membuat dan melampirkan kebijakan sumber daya API Gateway ke API

Agar pengguna dapat mengakses API Anda dengan memanggil layanan eksekusi API, Anda harus membuat kebijakan sumber daya API Gateway dan melampirkan kebijakan tersebut ke API. Saat Anda melampirkan kebijakan ke API Anda, kebijakan tersebut menerapkan izin dalam kebijakan ke metode di API. Jika memperbarui kebijakan sumber daya, Anda harus menerapkan API.

Topik

- [Prasyarat](#)
- [Melampirkan kebijakan sumber daya ke API Gateway API](#)
- [Memecahkan masalah kebijakan sumber daya Anda](#)

Prasyarat

Untuk memperbarui kebijakan sumber daya API Gateway, Anda memerlukan `apigateway:UpdateRestApiPolicy` izin dan `apigateway:PATCH` izin.

Untuk API Regional atau yang dioptimalkan tepi, Anda dapat melampirkan kebijakan sumber daya ke API saat Anda membuatnya, atau setelah diterapkan. Untuk API pribadi, Anda tidak dapat

menerapkan API tanpa kebijakan sumber daya. Untuk informasi selengkapnya, lihat [the section called “REST pribadi APIs”](#).

Melampirkan kebijakan sumber daya ke API Gateway API

Prosedur berikut menunjukkan cara melampirkan kebijakan sumber daya ke API Gateway API.

Konsol Manajemen AWS

Untuk melampirkan kebijakan sumber daya ke API Gateway API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Pilih Buat kebijakan.
5. (Opsional) Pilih templat untuk menghasilkan contoh kebijakan.

Dalam contoh kebijakan, placeholder dilampirkan dalam double curly braces ().

"`{{placeholder}}`" Ganti masing-masing placeholder, termasuk kawat gigi keriting, dengan informasi yang diperlukan.

6. Jika Anda tidak menggunakan salah satu contoh templat, masukkan kebijakan sumber daya Anda.
7. Pilih Simpan perubahan.

Jika API telah digunakan sebelumnya di konsol API Gateway, Anda harus menerapkannya kembali agar kebijakan sumber daya diterapkan.

AWS CLI

Untuk menggunakan AWS CLI untuk membuat API baru dan melampirkan kebijakan sumber daya ke dalamnya, gunakan [create-rest-api](#) perintah berikut:

```
aws apigateway create-rest-api \  
  --name "api-name" \  
  --policy "{\">jsonEscapedPolicyDocument\"}"
```

Untuk menggunakan AWS CLI untuk melampirkan kebijakan sumber daya ke API yang ada, gunakan [update-rest-api](#) perintah berikut:

```
aws apigateway update-rest-api \
  --rest-api-id api-id \
  --patch-operations op=replace,path=/
policy,value='{"jsonEscapedPolicyDocument"}'
```

Anda juga dapat melampirkan kebijakan sumber daya Anda sebagai `policy.json` file terpisah dan memasukkannya ke dalam `create-rest-api` perintah Anda. `create-rest-api` Perintah berikut membuat API baru dengan kebijakan sumber daya:

```
aws apigateway create-rest-api \
  --name "api-name" \
  --policy file://policy.json
```

`policy.json` adalah kebijakan sumber daya API Gateway, seperti [the section called “Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber”](#).

AWS CloudFormation

Anda dapat menggunakan CloudFormation untuk membuat API dengan kebijakan sumber daya. Contoh berikut membuat REST API dengan contoh kebijakan sumber daya, [the section called “Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber”](#).

```
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
```

Memecahkan masalah kebijakan sumber daya Anda

Panduan pemecahan masalah berikut dapat membantu menyelesaikan masalah dengan kebijakan sumber daya Anda.

API saya mengembalikan {"Message": "User: anonymous tidak diizinkan untuk melakukan: execute-api:invoke on resource: arn:aws:execute-api:us-east-1: *****/****/****/ "}

Dalam kebijakan sumber daya Anda, jika Anda menetapkan Principal ke AWS prinsipal, seperti berikut ini:

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"
      ]
    }
  ]
}
```

Anda harus menggunakan AWS_IAM otorisasi untuk setiap metode di API Anda, atau API Anda mengembalikan pesan kesalahan sebelumnya. Untuk petunjuk selengkapnya tentang cara mengaktifkan AWS_IAM otorisasi untuk suatu metode, lihat [the section called “Metode”](#).

Kebijakan sumber daya saya tidak diperbarui

Jika memperbarui kebijakan sumber daya setelah API dibuat, Anda harus menerapkan API untuk menyebarkan perubahan setelah Anda melampirkan kebijakan yang diperbarui. Memperbarui atau menyimpan kebijakan saja tidak akan mengubah perilaku runtime API. Untuk informasi selengkapnya tentang penerapan API Anda, lihat [the section called “Menyebarkan REST APIs”](#).

Kebijakan sumber daya saya mengembalikan kesalahan berikut: Dokumen kebijakan tidak valid. Silakan periksa sintaks kebijakan dan pastikan bahwa Prinsipal valid.

Untuk mengatasi masalah kesalahan ini, sebaiknya periksa sintaks kebijakan terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway”](#). Kami juga menyarankan Anda memeriksa bahwa semua prinsipal yang ditentukan valid dan belum dihapus.

Selain itu, jika API Anda berada di [Wilayah keikutsertaan](#), verifikasi bahwa semua akun dalam kebijakan sumber daya mengaktifkan Region.

AWS kunci kondisi yang dapat digunakan dalam kebijakan sumber daya API Gateway

Tabel berikut berisi kunci AWS kondisi yang dapat digunakan dalam kebijakan sumber daya APIs di API Gateway untuk setiap jenis otorisasi.

Untuk informasi selengkapnya tentang kunci AWS kondisi, lihat [Kunci Konteks Kondisi AWS Global](#).

Kunci kondisi	Kriteria	KebutuhanAuthN?	Jenis otorisasi
<code>aws:CurrentTime</code>	Tidak ada	Tidak	Semua
<code>aws:EpochTime</code>	Tidak ada	Tidak	Semua
<code>aws:Token IssueTime</code>	Kunci hanya ada dalam permintaan yang ditandatangani menggunakan kredensial keamanan sementara.	Ya	IAM
<code>aws:Multi FactorAuth Present</code>	Kunci hanya ada dalam permintaan yang ditandatangani menggunakan kredensial keamanan sementara.	Ya	IAM

Kunci kondisi	Kriteria	KebutuhanAuthN?	Jenis otorisasi
<code>aws:MultiFactorAuthAge</code>	Kunci hadir hanya jika MFA hadir dalam permintaan.	Ya	IAM
<code>aws:PrincipalAccount</code>	Tidak ada	Ya	IAM
<code>aws:PrincipalArn</code>	Tidak ada	Ya	IAM
<code>aws:PrincipalOrgID</code>	Kunci ini disertakan dalam konteks permintaan hanya jika kepala sekolah adalah anggota organisasi.	Ya	IAM
<code>aws:PrincipalOrgPaths</code>	Kunci ini disertakan dalam konteks permintaan hanya jika kepala sekolah adalah anggota organisasi.	Ya	IAM
<code>aws:PrincipalTag</code>	Kunci ini disertakan dalam konteks permintaan jika prinsipal menggunakan pengguna IAM dengan tag terlampir. Ini disertakan untuk prinsipal yang menggunakan peran IAM dengan tag yang disematkan atau tag sesi.	Ya	IAM

Kunci kondisi	Kriteria	KebutuhanAuthN?	Jenis otorisasi
aws:PrincipalType	Tidak ada	Ya	IAM
aws:Referer	Kunci hadir hanya jika nilai disediakan oleh pemanggil di header HTTP.	Tidak	Semua
aws:SecureTransport	Tidak ada	Tidak	Semua
aws:SourceArn	Tidak ada	Tidak	Semua
aws:SourceIp	Tidak ada	Tidak	Semua
aws:SourceVpc	Kunci ini hanya dapat digunakan untuk pribadi APIs.	Tidak	Semua
aws:SourceVpce	Kunci ini hanya dapat digunakan untuk pribadi APIs.	Tidak	Semua
aws:VpcSourceIp	Kunci ini hanya dapat digunakan untuk pribadi APIs.	Tidak	Semua
aws:UserAgent	Kunci hadir hanya jika nilai disediakan oleh pemanggil di header HTTP.	Tidak	Semua
aws:userid	Tidak ada	Ya	IAM
aws:username	Tidak ada	Ya	IAM

Kontrol akses ke REST API dengan izin IAM

Anda mengontrol akses ke Amazon API Gateway API dengan [izin IAM](#) dengan mengontrol akses ke dua proses komponen API Gateway berikut:

- Untuk membuat, menerapkan, dan mengelola API di API Gateway, Anda harus memberikan izin pengembang API untuk melakukan tindakan yang diperlukan yang didukung oleh komponen manajemen API API Gateway.
- Untuk memanggil API yang diterapkan atau menyegarkan cache API, Anda harus memberikan izin pemanggil API untuk melakukan tindakan IAM yang diperlukan yang didukung oleh komponen eksekusi API API Gateway.

Kontrol akses untuk dua proses melibatkan model izin yang berbeda, dijelaskan selanjutnya.

Model izin API Gateway untuk membuat dan mengelola API

[Untuk mengizinkan pengembang API membuat dan mengelola API di API Gateway, Anda harus membuat kebijakan izin IAM yang memungkinkan pengembang API tertentu untuk membuat, memperbarui, menerapkan, melihat, atau menghapus entitas API yang diperlukan.](#) Anda melampirkan kebijakan izin ke pengguna, peran, atau grup.

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Buat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) dalam Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Buat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang cara menggunakan model izin ini, lihat [the section called “Kebijakan berbasis identitas API Gateway”](#).

Model izin API Gateway untuk menjalankan API

Untuk mengizinkan pemanggil API menjalankan API atau menyegarkan caching, Anda harus membuat kebijakan IAM yang mengizinkan pemanggil API tertentu untuk menjalankan metode API yang memungkinkan otentikasi pengguna diaktifkan. Pengembang API menetapkan `authorizationType` properti metode `AWS_IAM` agar pemanggil mengirimkan kredensi pengguna untuk diautentikasi. API Gateway mendukung Signature Version 4a (Sigv4a) dan Signature Version 4 (SigV4) untuk mengautentikasi kredensi pengguna. Untuk informasi selengkapnya, lihat [Versi AWS Tanda Tangan 4](#). Kemudian, Anda melampirkan kebijakan ke pengguna, peran, atau grup.

[Dalam pernyataan kebijakan izin IAM ini, Resource elemen IAM berisi daftar metode API yang diterapkan yang diidentifikasi oleh kata kerja HTTP dan jalur sumber daya API Gateway yang diberikan.](#) Action Elemen IAM berisi tindakan eksekusi API Gateway API yang diperlukan. Tindakan ini mencakup `execute-api:Invoke` atau `execute-api:InvalidCache`, di mana `execute-api` menunjuk komponen eksekusi API yang mendasari API Gateway.

Untuk informasi selengkapnya tentang cara menggunakan model izin ini, lihat [Kontrol akses untuk menjalankan API](#).

Ketika API terintegrasi dengan AWS layanan (misalnya, AWS Lambda) di bagian belakang, API Gateway juga harus memiliki izin untuk mengakses AWS sumber daya terintegrasi (misalnya, menjalankan fungsi Lambda) atas nama pemanggil API. Untuk memberikan izin ini, buat peran IAM dari AWS layanan untuk jenis API Gateway. Saat Anda membuat peran ini di konsol Manajemen IAM, peran yang dihasilkan ini berisi kebijakan kepercayaan IAM berikut yang mendeklarasikan API Gateway sebagai entitas tepercaya yang diizinkan untuk mengambil peran:

JSON

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

```
        "Action": "sts:AssumeRole"
      }
    ]
  }
}
```

Jika Anda membuat peran IAM dengan memanggil [perintah create-role](#) CLI atau metode SDK yang sesuai, Anda harus memberikan kebijakan kepercayaan di atas sebagai parameter input. `assume-role-policy-document` Jangan mencoba membuat kebijakan semacam itu secara langsung di konsol Manajemen IAM atau memanggil perintah AWS CLI [create-policy](#) atau metode SDK yang sesuai.

Agar API Gateway memanggil AWS layanan terintegrasi, Anda juga harus melampirkan kebijakan izin IAM yang sesuai peran ini untuk memanggil layanan terintegrasi AWS . Misalnya, untuk memanggil fungsi Lambda, Anda harus menyertakan kebijakan izin IAM berikut dalam peran IAM:

JSON

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

Perhatikan bahwa Lambda mendukung kebijakan akses berbasis sumber daya, yang menggabungkan kebijakan kepercayaan dan izin. Saat mengintegrasikan API dengan fungsi Lambda menggunakan konsol API Gateway, Anda tidak diminta untuk menyetel peran IAM ini secara eksplisit, karena konsol menetapkan izin berbasis sumber daya pada fungsi Lambda untuk Anda, dengan persetujuan Anda.

Note

Untuk memberlakukan kontrol akses ke AWS layanan, Anda dapat menggunakan model izin berbasis pemanggil, di mana kebijakan izin langsung dilampirkan ke pengguna atau

grup pemanggil, atau model izin berbasis peran, di mana kebijakan izin dilampirkan ke peran IAM yang dapat diasumsikan oleh API Gateway. Kebijakan izin mungkin berbeda dalam kedua model. Misalnya, kebijakan berbasis pemanggil memblokir akses sementara kebijakan berbasis peran mengizinkannya. Anda dapat memanfaatkan ini untuk mengharuskan pengguna mengakses AWS layanan hanya melalui API Gateway API.

Kontrol akses untuk menjalankan API

Di bagian ini, Anda mempelajari tentang model izin untuk mengontrol akses ke API Anda menggunakan izin IAM. Ketika otorisasi IAM diaktifkan, klien harus menggunakan Signature Version 4a (Sigv4a) dan Sigv4a Versi Tanda Tangan 4 (SigV4) untuk menandatangani permintaan mereka dengan kredensi. AWS Untuk informasi selengkapnya, lihat [Versi AWS Tanda Tangan 4](#).

Pada bagian ini, kami menampilkan templat pernyataan kebijakan IAM dan referensi pernyataan kebijakan. Referensi pernyataan kebijakan mencakup format Action dan Resource bidang yang terkait dengan layanan eksekusi API. Gunakan referensi ini untuk membuat pernyataan kebijakan IAM Anda. Saat membuat pernyataan kebijakan IAM, Anda mungkin perlu mempertimbangkan bagaimana kebijakan sumber daya API Gateway memengaruhi alur kerja otorisasi. Untuk informasi selengkapnya, lihat [the section called “Bagaimana kebijakan sumber daya memengaruhi alur kerja otorisasi”](#).

Untuk pribadi APIs, Anda harus menggunakan kombinasi kebijakan sumber daya API Gateway dan kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat topik berikut:

- [the section called “Menggunakan kebijakan sumber daya API Gateway”](#)
- [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#)

Kontrol siapa yang dapat memanggil metode API Gateway API dengan kebijakan IAM

Untuk mengontrol siapa yang dapat atau tidak dapat memanggil API yang diterapkan dengan izin IAM, buat dokumen kebijakan IAM dengan izin yang diperlukan. Template untuk dokumen kebijakan semacam itu ditampilkan sebagai berikut.

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"
    ]
  }
]
}

```

Di sini, *Permission* akan diganti oleh Allow atau Deny tergantung pada apakah Anda ingin memberikan atau mencabut izin yang disertakan. *Execution-operation* akan digantikan oleh operasi yang didukung oleh layanan eksekusi API. *METHOD_HTTP_VERB* singkatan dari kata kerja HTTP yang didukung oleh sumber daya yang ditentukan. *Resource-path* adalah placeholder untuk jalur URL dari [Resource](#) instance API yang diterapkan yang mendukung hal tersebut. *METHOD_HTTP_VERB* Untuk informasi selengkapnya, lihat [Referensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway](#).

Note

Agar kebijakan IAM efektif, Anda harus mengaktifkan autentikasi IAM pada metode API dengan menyetel properti `AWS_IAM` metode. [authorizationType](#) Gagal melakukannya akan membuat metode API ini dapat diakses publik.

Misalnya, untuk memberikan izin kepada pengguna untuk melihat daftar hewan peliharaan yang diekspos oleh API tertentu, tetapi untuk menolak izin pengguna untuk menambahkan hewan peliharaan ke daftar, Anda dapat menyertakan pernyataan berikut dalam kebijakan IAM:

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"
    ]
}
]
}

```

Untuk memberikan izin kepada pengguna untuk melihat hewan peliharaan tertentu yang diekspos oleh API yang dikonfigurasi sebagai GET /pets/{*petId*}, Anda dapat menyertakan pernyataan berikut dalam kebijakan IAM:

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"
            ]
        }
    ]
}

```

Referensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway

Informasi berikut menjelaskan format Action and Resource dari pernyataan kebijakan IAM tentang izin akses untuk menjalankan API.

Format tindakan izin untuk menjalankan API di API Gateway

ActionEkspresi pelaksana API memiliki format umum berikut:

```
execute-api:action
```

di *action* mana tindakan pelaksana API yang tersedia:

- *, yang mewakili semua tindakan berikut.
- Invoke, digunakan untuk memanggil API atas permintaan klien.
- InvalidateCache, digunakan untuk membatalkan cache API atas permintaan klien.

Format sumber daya izin untuk menjalankan API di API Gateway

ResourceEkspresi pelaksana API memiliki format umum berikut:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-specifier
```

di mana:

- *region* adalah AWS wilayah (seperti **us-east-1** atau * untuk semua AWS wilayah) yang sesuai dengan API yang diterapkan untuk metode tersebut.
- *account-id* adalah Id AWS akun 12 digit dari pemilik REST API.
- *api-id* adalah pengidentifikasi API Gateway yang telah ditetapkan ke API untuk metode tersebut.
- *stage-name* adalah nama tahap yang terkait dengan metode.
- *HTTP-VERB* adalah kata kerja HTTP untuk metode ini. Ini bisa menjadi salah satu dari yang berikut: GET, POST, PUT, DELETE, PATCH.
- *resource-path-specifier* adalah jalan menuju metode yang diinginkan.

Note

Jika Anda menentukan wildcard (*), Resource ekspresi akan menerapkan wildcard ke ekspresi lainnya.

Beberapa contoh ekspresi sumber daya meliputi:

- **arn:aws:execute-api:::*:** untuk jalur sumber daya apa pun di tahap apa pun, untuk API apa pun di AWS wilayah mana pun.
- **arn:aws:execute-api:us-east-1:::** untuk jalur sumber daya apa pun di tahap apa pun, untuk API apa pun di AWS wilayah us-east-1.
- **arn:aws:execute-api:us-east-1:::api-id/*** untuk jalur sumber daya apa pun di tahap apa pun, untuk API dengan pengenal *api-id* di AWS wilayah us-east-1.
- **arn:aws:execute-api:us-east-1:::api-id/test/*** untuk jalur sumber daya apa pun pada tahap test, untuk API dengan pengenal *api-id* di AWS wilayah us-east-1.

Untuk mempelajari selengkapnya, lihat [Referensi API Gateway Amazon Resource Name \(ARN\)](#).

Contoh kebijakan IAM untuk izin eksekusi API

Untuk model izin dan informasi latar belakang lainnya, lihat [Kontrol akses untuk menjalankan API](#).

Pernyataan kebijakan berikut memberikan izin kepada pengguna untuk memanggil metode POST apa pun di sepanjang jalur `mydemoresource`, pada tahap `test`, untuk API dengan pengenal `123456789`, dengan asumsi API yang sesuai telah diterapkan ke wilayah AWS us-east-1:

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/*"
  ]
}
]
}

```

Contoh pernyataan kebijakan berikut memberikan izin kepada pengguna untuk memanggil metode apa pun di jalur sumber daya `petstorewalkthrough/pets`, dalam tahap apa pun, untuk API dengan pengenal `a123456789`, di AWS wilayah mana pun di mana API terkait telah digunakan:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}

```

Menggunakan kebijakan titik akhir VPC secara pribadi APIs di API Gateway

Untuk meningkatkan keamanan API pribadi Anda, Anda dapat membuat kebijakan titik akhir VPC. Kebijakan titik akhir VPC adalah kebijakan sumber daya IAM yang Anda lampirkan ke titik akhir VPC. Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan VPC Endpoints](#).

Anda mungkin ingin membuat kebijakan titik akhir VPC untuk melakukan tugas-tugas berikut.

- Izinkan hanya organisasi atau sumber daya tertentu untuk mengakses titik akhir VPC Anda dan menjalankan API Anda.

- Gunakan satu kebijakan dan hindari kebijakan berbasis sesi atau berbasis peran untuk mengontrol lalu lintas ke API Anda.
- Kencangkan perimeter keamanan aplikasi Anda saat bermigrasi dari tempat ke AWS lokasi.

Pertimbangan kebijakan titik akhir VPC

Berikut ini adalah pertimbangan untuk kebijakan titik akhir VPC Anda:

- Identitas invoker dievaluasi berdasarkan nilai `Authorization` header. Kebijakan titik akhir VPC dievaluasi terlebih dahulu, lalu API Gateway mengevaluasi permintaan, berdasarkan jenis otorisasi yang dikonfigurasi pada permintaan metode. Tabel berikut menunjukkan bagaimana kebijakan titik akhir VPC dievaluasi berdasarkan isi nilai header. `Authorization`

Isi dari nilai Authorization header	Bagaimana kebijakan titik akhir VPC dievaluasi
Tidak ada konten	Pemanggil dievaluasi sebagai pengguna anonim
Tanda tangan SigV4 atau Sigv4a yang valid	Pemanggil dievaluasi sebagai identitas IAM yang diautentikasi dari tanda tangan
Tanda tangan SigV4 atau Sigv4a tidak valid	API Gateway menolak akses
Informasi otorisasi non-SIGV4 seperti token pembawa	Pemanggil dievaluasi sebagai pengguna anonim

- [Jika kontrol akses Anda bergantung pada penggunaan token pembawa, seperti Lambda atau otorisasi Amazon Cognito, Anda dapat mengontrol perimeter keamanan Anda dengan menggunakan properti sumber daya.](#)
- Jika kontrol otorisasi Anda menggunakan otorisasi IAM, Anda dapat mengontrol perimeter keamanan Anda dengan menggunakan [properti sumber daya](#) dan [properti](#) prinsipal.
- Kebijakan titik akhir VPC dapat digunakan bersama dengan kebijakan sumber daya API Gateway. Kebijakan sumber daya API Gateway menentukan prinsipal mana yang dapat mengakses API. Kebijakan endpoint menentukan siapa yang dapat mengakses VPC dan mana yang APIs dapat dipanggil dari titik akhir VPC. API pribadi Anda memerlukan kebijakan sumber daya tetapi Anda tidak perlu membuat kebijakan titik akhir VPC kustom.

Contoh kebijakan titik akhir VPC

Anda dapat membuat kebijakan untuk titik akhir Amazon Virtual Private Cloud untuk Amazon API Gateway yang dapat Anda tentukan berikut ini.

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat memiliki tindakan yang dilakukan pada mereka.

Ini mungkin tergantung pada isi header otorisasi Anda. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan kebijakan titik akhir VPC”](#). Untuk kebijakan contoh tambahan, lihat [Contoh kebijakan perimeter data](#) di GitHub situs web.

Untuk melampirkan kebijakan ke titik akhir VPC, Anda harus menggunakan konsol VPC. Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan VPC Endpoints](#).

Contoh 1: Kebijakan titik akhir VPC yang memberikan akses ke dua APIs

Contoh kebijakan berikut memberikan akses ke hanya dua spesifik APIs melalui titik akhir VPC yang dilampirkan kebijakan tersebut.

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/*"
      ]
    }
  ]
}
```

Contoh 2: Kebijakan titik akhir VPC yang memberikan akses ke metode GET

Contoh kebijakan berikut memberi pengguna akses ke GET metode untuk API tertentu melalui titik akhir VPC tempat kebijakan dilampirkan.

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/*"
      ]
    }
  ]
}
```

Contoh 3: Kebijakan titik akhir VPC yang memberikan akses pengguna tertentu ke API tertentu

Contoh kebijakan berikut memberikan akses pengguna tertentu ke API tertentu melalui titik akhir VPC tempat kebijakan dilampirkan.

Dalam hal ini, karena kebijakan membatasi akses ke prinsipal IAM tertentu, Anda harus menyetel metode ke `authorizationType`. `AWS_IAM` `NONE`

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/*"
    ]
  }
}

```

Contoh 4: Kebijakan titik akhir VPC yang memberikan pengguna akses ke nama domain khusus tertentu dan setiap API yang dipetakan ke domain

Contoh kebijakan berikut memberi pengguna akses ke nama domain khusus khusus untuk pribadi APIs melalui titik akhir VPC tempat kebijakan dilampirkan. Dengan kebijakan ini, selama pengguna telah membuat asosiasi akses nama domain antara titik akhir VPC dan nama domain kustom dan diberikan akses untuk memanggil nama domain kustom dan API pribadi apa pun yang dipetakan ke nama domain kustom, pengguna dapat memanggil apa pun APIs yang dipetakan ke nama domain kustom ini. Untuk informasi selengkapnya, lihat [the section called “Nama domain khusus untuk pribadi 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",
    }
  }
]
}

```

Contoh 5: Kebijakan titik akhir VPC yang memberikan atau menolak akses ke sumber daya spesifik dan domain APIs

Contoh kebijakan berikut memberi pengguna akses ke sumber daya tertentu APIs dan domain. Dengan kebijakan ini, selama pengguna telah membuat asosiasi akses nama domain antara titik akhir VPC dan nama domain kustom dan diberikan akses untuk memanggil nama domain kustom dan API pribadi apa pun yang dipetakan ke nama domain kustom, pengguna dapat memanggil sumber daya pribadi dan domain yang diizinkan. APIs Untuk informasi selengkapnya, lihat [the section called “Nama domain khusus untuk pribadi 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/*"
    ]
  }
]
}

```

Contoh 6: Kebijakan titik akhir VPC yang memberikan atau menolak akses oleh kepala sekolah dan sumber daya milik suatu organisasi

Contoh kebijakan berikut memberikan akses ke kepala sekolah dan sumber daya milik organisasi.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringEquals": {
          "aws:ResourceOrgID": "o-abcd1234",
          "aws:PrincipalOrgID": "o-abcd1234"
        }
      },
      "Action": "*",
      "Resource": "*",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources"
    }
  ]
}

```

Gunakan tag untuk mengontrol akses ke REST APIs di API Gateway

Izin untuk mengakses REST APIs dapat disetel dengan baik menggunakan kontrol akses berbasis atribut dalam kebijakan IAM.

Lihat informasi yang lebih lengkap di [the section called “Kontrol Akses Berbasis Atribut”](#).

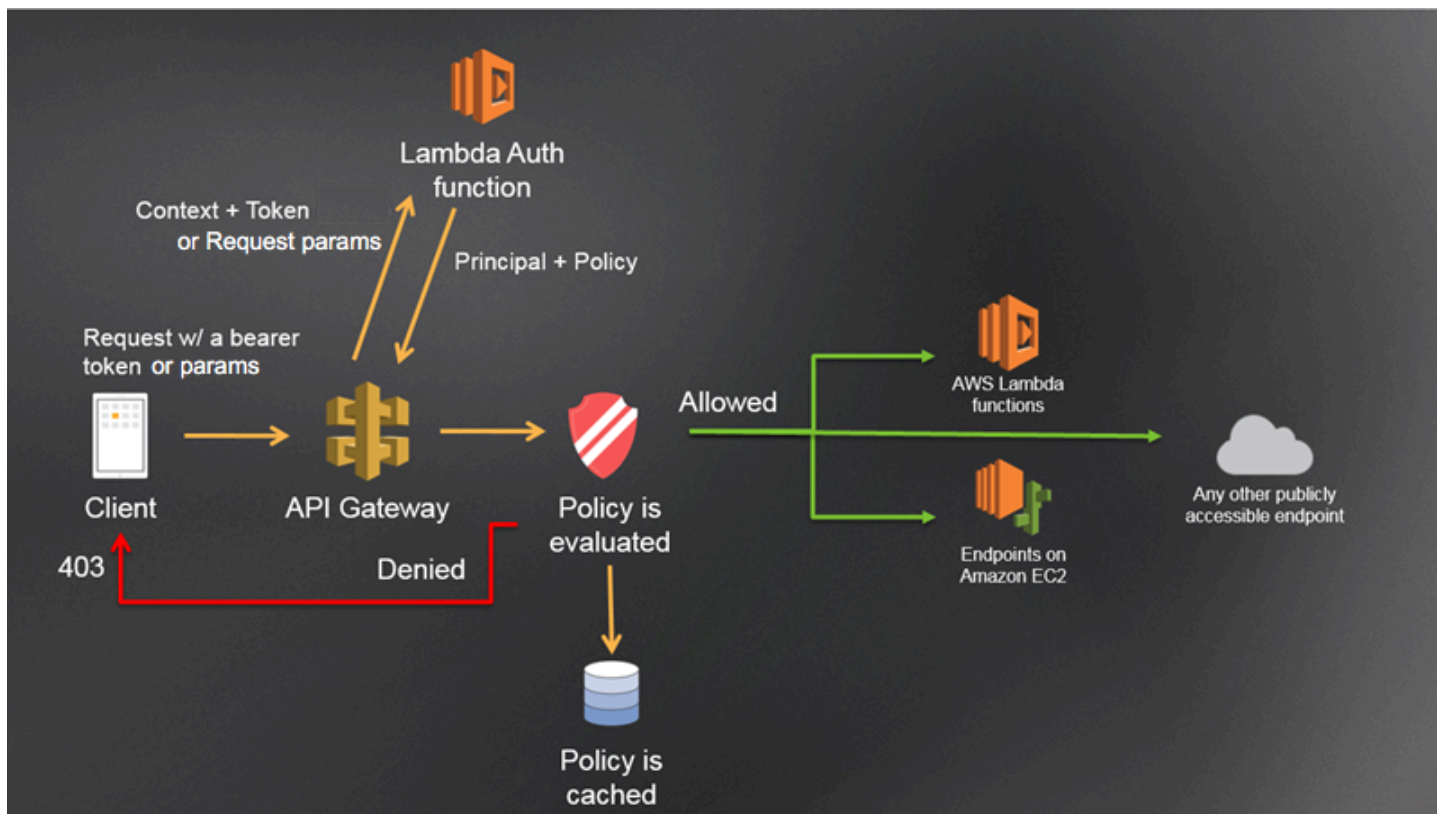
Gunakan otorisasi API Gateway Lambda

Gunakan Lambda Authorizer (sebelumnya dikenal sebagai otorisasi kustom) untuk mengontrol akses ke API Anda. Saat klien membuat permintaan ke metode API Anda, API Gateway memanggil otorisasi Lambda Anda. Authorizer Lambda mengambil identitas pemanggil sebagai input dan mengembalikan kebijakan IAM sebagai output.

Gunakan otorisasi Lambda untuk menerapkan skema otorisasi khusus. Skema Anda dapat menggunakan parameter permintaan untuk menentukan identitas pemanggil atau menggunakan strategi otentikasi token pembawa seperti OAuth atau SALL. Buat otorisasi Lambda di konsol API API Gateway REST API, menggunakan AWS CLI, atau SDK. AWS

Alur kerja otorisasi otorisasi Lambda

Diagram berikut menunjukkan alur kerja otorisasi untuk otorisasi Lambda.



Alur kerja otorisasi API Gateway Lambda

1. Klien memanggil metode pada API Gateway API, meneruskan token pembawa atau parameter permintaan.
2. API Gateway memeriksa apakah permintaan metode dikonfigurasi dengan otorisasi Lambda. Jika ya, API Gateway memanggil fungsi Lambda.
3. Fungsi Lambda mengautentikasi pemanggil. Fungsi ini dapat mengautentikasi dengan cara-cara berikut:
 - Dengan memanggil OAuth penyedia untuk mendapatkan token OAuth akses.
 - Dengan memanggil penyedia SAFL untuk mendapatkan pernyataan SAFL.
 - Dengan membuat kebijakan IAM berdasarkan nilai parameter permintaan.
 - Dengan mengambil kredensi dari database.
4. Fungsi Lambda mengembalikan kebijakan IAM dan pengidentifikasi utama. Jika fungsi Lambda tidak mengembalikan informasi itu, panggilan gagal.
5. API Gateway mengevaluasi kebijakan IAM.
 - Jika akses ditolak, API Gateway mengembalikan kode status HTTP yang sesuai, seperti 403 ACCESS_DENIED.
 - Jika akses diizinkan, API Gateway akan memanggil metode.

Jika Anda mengaktifkan caching otorisasi, API Gateway menyimpan kebijakan agar fungsi otorisasi Lambda tidak dipanggil lagi. Pastikan kebijakan Anda berlaku untuk semua sumber daya dan metode di seluruh API Anda.

Anda dapat menyesuaikan 403 ACCESS_DENIED atau tanggapan 401 UNAUTHORIZED gateway. Untuk mempelajari selengkapnya, lihat [the section called “Tanggapan Gateway”](#).

Memilih jenis otorisasi Lambda

Ada dua jenis otorisasi Lambda:

Minta otorisasi Lambda berbasis parameter (otorisasi) **REQUEST**

REQUESTAuthorizer menerima identitas pemanggil dalam kombinasi header, parameter string kueri, dan variabel [stageVariables](#). [\\$context](#) Anda dapat menggunakan REQUEST otorisasi untuk membuat kebijakan berbutir halus berdasarkan informasi dari beberapa sumber identitas, seperti variabel dan konteks. `$context.path` `$context.httpMethod`

Jika Anda mengaktifkan caching otorisasi untuk REQUEST otorisasi, API Gateway memverifikasi bahwa semua sumber identitas yang ditentukan ada dalam permintaan. Jika sumber identifikasi tertentu hilang, null, atau kosong, API Gateway mengembalikan respons 401 Unauthorized HTTP tanpa memanggil fungsi otorisasi Lambda. Ketika beberapa sumber identitas didefinisikan, mereka semua digunakan untuk mendapatkan kunci cache otorisasi, dengan urutan dipertahankan. Anda dapat menentukan kunci cache berbutir halus dengan menggunakan beberapa sumber identitas.

Jika Anda mengubah salah satu bagian kunci cache, dan menerapkan ulang API Anda, otorisasi akan membuang dokumen kebijakan yang di-cache dan membuat yang baru.

Jika Anda mematikan caching otorisasi untuk REQUEST otorisasi, API Gateway langsung meneruskan permintaan ke fungsi Lambda.

Pengotorisasi Lambda berbasis token (otorisasi) **TOKEN**

TOKENAuthorizer menerima identitas penelepon dalam token pembawa, seperti JSON Web Token (JWT) atau token. OAuth

Jika Anda mengaktifkan caching otorisasi untuk TOKEN otorisasi, nama header yang ditentukan dalam sumber token menjadi kunci cache.

Selain itu, Anda dapat menggunakan validasi token untuk memasukkan RegEx pernyataan. API Gateway melakukan validasi awal token input terhadap ekspresi ini dan memanggil fungsi otorisasi Lambda setelah validasi berhasil. Ini membantu mengurangi panggilan ke API Anda.

IdentityValidationExpressionProperti ini didukung hanya untuk TOKEN otorisasi. Untuk informasi selengkapnya, lihat [the section called “x-amazon-apigateway-authorizer”](#).

Note

Kami menyarankan Anda menggunakan REQUEST otorisasi untuk mengontrol akses ke API Anda. Anda dapat mengontrol akses ke API berdasarkan beberapa sumber identitas saat menggunakan REQUEST otorisasi, dibandingkan dengan satu sumber identitas saat menggunakan TOKEN otorisasi. Selain itu, Anda dapat memisahkan kunci cache menggunakan beberapa sumber identitas untuk REQUEST otorisasi.

Contoh fungsi Lambda **REQUEST** authorizer

Kode contoh berikut membuat fungsi otorisasi Lambda yang memungkinkan permintaan jika HeaderAuth1 header yang disediakan klien, parameter QueryString1 kueri, dan variabel tahap StageVar1 semua cocok dengan nilai yang ditentukan,, danheaderValue1, queryValue1 masing-masing. stageValue1

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)
```

Dalam contoh ini, fungsi Lambda Authorizer memeriksa parameter input dan bertindak sebagai berikut:

- Jika semua nilai parameter yang diperlukan cocok dengan nilai yang diharapkan, fungsi authorizer mengembalikan respons 200 OK HTTP dan kebijakan IAM yang terlihat seperti berikut, dan permintaan metode berhasil:

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/"
    }
  ]
}
```

- Jika tidak, fungsi authorizer mengembalikan respon 401 Unauthorized HTTP, dan permintaan metode gagal.

Selain mengembalikan kebijakan IAM, fungsi otorisasi Lambda juga harus mengembalikan pengenalan utama pemanggil. Secara opsional, ini dapat mengembalikan context objek yang berisi informasi tambahan yang dapat diteruskan ke backend integrasi. Untuk informasi selengkapnya, lihat [Keluaran dari otorisasi API Gateway Lambda](#).

Dalam kode produksi, Anda mungkin perlu mengautentikasi pengguna sebelum memberikan otorisasi. Anda dapat menambahkan logika otentikasi dalam fungsi Lambda dengan memanggil penyedia otentikasi seperti yang diarahkan dalam dokumentasi untuk penyedia tersebut.

Contoh fungsi Lambda **TOKEN** authorizer

Kode contoh berikut membuat fungsi otorisasi TOKEN Lambda yang memungkinkan pemanggil untuk memanggil metode jika nilai token yang disediakan klien adalah `allow`. Penelepon tidak diizinkan untuk memanggil permintaan jika nilai tokennya `deny`. Jika nilai token adalah `unauthorized` atau string kosong, fungsi authorizer mengembalikan 401 UNAUTHORIZED respons.

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
```

Dalam contoh ini, ketika API menerima permintaan metode, API Gateway meneruskan token sumber ke fungsi otorisasi Lambda ini di atribut `event.authorizationToken`. Fungsi otorisasi Lambda membaca token dan bertindak sebagai berikut:

- Jika nilai `tokenAllow`, fungsi authorizer mengembalikan respons 200 OK HTTP dan kebijakan IAM yang terlihat seperti berikut, dan permintaan metode berhasil:

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/"
    }
  ]
}
```

- Jika nilai `tokendeny`, fungsi `authorizer` mengembalikan respons `200 OK HTTP` dan kebijakan `Deny IAM` yang terlihat seperti berikut, dan permintaan metode gagal:

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

Di luar lingkungan pengujian, API Gateway mengembalikan respons `403 Forbidden HTTP` dan permintaan metode gagal.

- Jika nilai token `unauthorized` atau string kosong, fungsi `authorizer` mengembalikan respons `401 Unauthorized HTTP`, dan panggilan metode gagal.
- Jika token adalah hal lain, klien menerima `500 Invalid token` respons, dan panggilan metode gagal.

Selain mengembalikan kebijakan IAM, fungsi otorisasi Lambda juga harus mengembalikan pengenalan pemanggil. Secara opsional, ini dapat mengembalikan `context` objek yang berisi informasi tambahan yang dapat diteruskan ke backend integrasi. Untuk informasi selengkapnya, lihat [Keluaran dari otorisasi API Gateway Lambda](#).

Dalam kode produksi, Anda mungkin perlu mengautentikasi pengguna sebelum memberikan otorisasi. Anda dapat menambahkan logika otentikasi dalam fungsi Lambda dengan memanggil penyedia otentikasi seperti yang diarahkan dalam dokumentasi untuk penyedia tersebut.

Contoh tambahan fungsi otorisasi Lambda

Daftar berikut menunjukkan contoh tambahan fungsi otorisasi Lambda. Anda dapat membuat fungsi Lambda di akun yang sama, atau akun lain, dari tempat Anda membuat API.

Untuk contoh fungsi Lambda sebelumnya, Anda dapat menggunakan built-in [AWSLambdaBasicExecutionRole](#), karena fungsi ini tidak memanggil layanan lain AWS. Jika fungsi Lambda Anda memanggil AWS layanan lain, Anda harus menetapkan peran eksekusi IAM ke fungsi Lambda. Untuk membuat peran, ikuti instruksi dalam [Peran AWS Lambda Eksekusi](#).

Contoh tambahan fungsi otorisasi Lambda

- Untuk contoh aplikasi, lihat [Buka Perbankan Brasil - Sampel GitHub Otorisasi pada](#).
- Untuk contoh fungsi Lambda lainnya, lihat [aws-apigateway-lambda-authorizer-blueprints](#) on GitHub
- Anda dapat membuat otorisasi Lambda yang mengautentikasi pengguna menggunakan kumpulan pengguna Amazon Cognito dan mengotorisasi penelepon berdasarkan penyimpanan kebijakan menggunakan Izin Terverifikasi. Untuk informasi selengkapnya, lihat [the section called “Kontrol akses berdasarkan atribut identitas dengan Izin Terverifikasi”](#).
- Konsol Lambda menyediakan cetak biru Python, yang dapat Anda gunakan dengan memilih Gunakan cetak biru dan memilih cetak biru. `api-gateway-authorizer-python`

Konfigurasikan otorisasi Lambda API Gateway

Setelah membuat fungsi Lambda, Anda mengonfigurasi fungsi Lambda sebagai otorisasi untuk API Anda. Anda kemudian mengonfigurasi metode Anda untuk memanggil otorisasi Lambda Anda untuk menentukan apakah pemanggil dapat memanggil metode Anda. Anda dapat membuat fungsi Lambda di akun yang sama, atau akun lain, dari tempat Anda membuat API.

[Anda dapat menguji otorisasi Lambda Anda menggunakan alat bawaan di konsol API Gateway atau dengan menggunakan Postman](#). Untuk petunjuk tentang cara menggunakan Postman untuk menguji fungsi otorisasi Lambda Anda, lihat [the section called “Panggil API dengan otorisasi Lambda”](#)

Konfigurasikan otorisasi Lambda (konsol)

Prosedur berikut menunjukkan cara membuat otorisasi Lambda di konsol API Gateway REST API. Untuk mempelajari lebih lanjut tentang berbagai jenis otorisasi Lambda, lihat [the section called “Memilih jenis otorisasi Lambda”](#)

REQUEST authorizer

Untuk mengkonfigurasi otorisasi **REQUEST** Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pilih API, lalu pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk nama Authorizer, masukkan nama untuk otorisasi.
5. Untuk jenis Authorizer, pilih Lambda.
6. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi otorisasi Lambda Anda, lalu masukkan nama fungsi.
7. Biarkan peran panggilan Lambda kosong agar konsol API Gateway REST API menetapkan kebijakan berbasis sumber daya. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi otorisasi Lambda. Anda juga dapat memilih untuk memasukkan nama peran IAM untuk mengizinkan API Gateway menjalankan fungsi otorisasi Lambda. Untuk peran contoh, lihat [Buat peran IAM yang dapat diasumsikan](#).
8. Untuk payload acara Lambda, pilih Minta.
9. Untuk tipe sumber Identity, pilih tipe parameter. Jenis parameter yang didukung adalah `Header`, `Query string`, `Stage variable`, dan `Context`. Untuk menambahkan lebih banyak sumber identitas, pilih Tambah parameter.
10. Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan.

Jika Anda mengaktifkan caching, otorisasi Anda harus mengembalikan kebijakan yang berlaku untuk semua metode di seluruh API. Untuk menegakkan kebijakan khusus metode, gunakan variabel konteks dan `$context.path` `$context.httpMethod`

11. Pilih Buat Authorizer.

TOKEN authorizer

Untuk mengkonfigurasi otorisasi **TOKEN** Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API, lalu pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk nama Authorizer, masukkan nama untuk otorisasi.
5. Untuk jenis Authorizer, pilih Lambda.

6. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi otorisasi Lambda Anda, lalu masukkan nama fungsi.
 7. Biarkan peran panggilan Lambda kosong agar konsol API Gateway REST API menetapkan kebijakan berbasis sumber daya. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi otorisasi Lambda. Anda juga dapat memilih untuk memasukkan nama peran IAM untuk mengizinkan API Gateway menjalankan fungsi otorisasi Lambda. Untuk peran contoh, lihat [Buat peran IAM yang dapat diasumsikan](#).
 8. Untuk payload acara Lambda, pilih Token.
 9. Untuk sumber Token, masukkan nama header yang berisi token otorisasi. Penelepon harus menyertakan header nama ini untuk mengirim token otorisasi ke otorisasi Lambda.
 10. (Opsional) Untuk validasi Token, masukkan RegEx pernyataan. API Gateway melakukan validasi awal token input terhadap ekspresi ini dan memanggil authorizer setelah validasi berhasil.
 11. Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, nama header yang ditentukan dalam sumber Token menjadi kunci cache. Saat caching kebijakan diaktifkan, Anda dapat mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan.
- Jika Anda mengaktifkan caching, otorisasi Anda harus mengembalikan kebijakan yang berlaku untuk semua metode di seluruh API. Untuk menerapkan kebijakan khusus metode, Anda dapat menonaktifkan caching Otorisasi.
12. Pilih Buat Authorizer.

Setelah Anda membuat otorisasi Lambda Anda, Anda dapat mengujinya. Prosedur berikut menunjukkan cara menguji otorisasi Lambda Anda.

REQUEST authorizer

Untuk menguji otorisasi **REQUEST** Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama otorisasi Anda.
3. Di bawah Pengotorisasi uji, masukkan nilai untuk sumber identitas Anda.

Jika Anda menggunakan [the section called “Contoh fungsi Lambda REQUEST authorizer”](#), lakukan hal berikut:

- a. Pilih Header dan enter **headerValue1**, lalu pilih Tambah parameter.
- b. Di bawah Jenis sumber identitas, pilih String kueri dan masukkan **queryValue1**, lalu pilih Tambah parameter.
- c. Di bawah Identity source type, pilih Stage variable dan enter **stageValue1**.

Anda tidak dapat memodifikasi variabel konteks untuk pemanggilan pengujian, tetapi Anda dapat memodifikasi template peristiwa pengujian API Gateway Authorizer untuk fungsi Lambda Anda. Kemudian, Anda dapat menguji fungsi otorisasi Lambda Anda dengan variabel konteks yang dimodifikasi. Untuk informasi selengkapnya, lihat [Menguji fungsi Lambda di konsol](#) di Panduan AWS Lambda Pengembang.

4. Pilih Test Authorizer.

TOKEN authorizer

Untuk menguji otorisasi **TOKEN** Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama otorisasi Anda.
3. Di bawah Pengotorisasi uji, masukkan nilai untuk token Anda.

Jika Anda menggunakan [the section called “Contoh fungsi Lambda TOKEN authorizer”](#), lakukan hal berikut:

- Untuk AuthorizationToken, masukkan. **allow**
4. Pilih Test Authorizer.

Jika otorisasi Lambda Anda berhasil menolak permintaan di lingkungan pengujian, pengujian akan merespons dengan respons HTTP. 200 OK Namun, di luar lingkungan pengujian, API Gateway mengembalikan respons 403 Forbidden HTTP dan permintaan metode gagal.

Konfigurasi otorisasi Lambda ()AWS CLI

Perintah [create-authorizer](#) berikut menunjukkan untuk membuat authorizer Lambda menggunakan AWS CLI

REQUEST authorizer

Perintah [create-authorizer](#) berikut membuat REQUEST authorizer dan menggunakan variabel Authorizer header dan accountId konteks sebagai sumber identitas:

```
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

Perintah [create-authorizer](#) berikut membuat TOKEN authorizer dan menggunakan Authorization header sebagai sumber identitas:

```
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
```

Setelah Anda membuat otorisasi Lambda Anda, Anda dapat mengujinya. [test-invoke-authorizer](#)Perintah berikut menguji otorisasi Lambda:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \  
  --authorizer-id efg1234 \  
  --headers Authorization='Value'
```

Konfigurasi metode untuk menggunakan otorisasi Lambda (konsol)

Setelah mengonfigurasi otorisasi Lambda, Anda harus melampirkannya ke metode untuk API Anda. Jika otorisasi Anda menggunakan caching otorisasi, pastikan Anda memperbarui kebijakan untuk mengontrol akses untuk metode tambahan.

Untuk mengonfigurasi metode API untuk menggunakan otorisasi Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API.
3. Pilih Sumber Daya, lalu pilih metode baru atau pilih metode yang ada.
4. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
5. Untuk Authorizer, dari menu dropdown, pilih Lambda Authorizer yang baru saja Anda buat.
6. (Opsional) Jika Anda ingin meneruskan token otorisasi ke backend, pilih header permintaan HTTP. Pilih Tambahkan header, lalu tambahkan nama header otorisasi. Untuk Nama, masukkan nama header yang cocok dengan nama sumber Token yang Anda tentukan saat membuat otorisasi Lambda untuk API. Langkah ini tidak berlaku untuk REQUEST otorisasi.
7. Pilih Simpan.
8. Pilih Deploy API untuk menerapkan API ke panggung. Untuk REQUEST otorisasi yang menggunakan variabel tahap, Anda juga harus menentukan variabel tahap yang diperlukan dan menentukan nilainya saat berada di halaman Tahapan.

Konfigurasi metode untuk menggunakan otorisasi Lambda (AWS CLI)

Setelah mengonfigurasi otorisasi Lambda, Anda harus melampirkannya ke metode untuk API Anda. Anda dapat membuat metode baru atau menggunakan operasi tambalan untuk melampirkan otorisasi ke metode yang ada. Jika otorisasi Anda menggunakan caching otorisasi, pastikan Anda memperbarui kebijakan untuk mengontrol akses untuk metode tambahan.

Perintah [put-method](#) berikut membuat metode baru yang menggunakan authorizer Lambda:

```
aws apigateway put-method --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --authorization-type CUSTOM \  
  --authorizer-id efg1234
```

Perintah [update-method](#) berikut memperbarui metode yang ada untuk menggunakan Lambda authorizer:

```
aws apigateway update-method \  
  --rest-api-id 1234123412 \  
  --resource-id a1b2c3
```

```
--resource-id a1b2c3 \
--http-method PUT \
--patch-operations op="replace",path="/authorizationType",value="CUSTOM"
op="replace",path="/authorizerId",value="efg1234"
```

Masukan ke otorisasi Lambda API Gateway

Bagian berikut menjelaskan format input dari API Gateway ke otorisasi Lambda.

TOKEN format masukan

Untuk Authorizer Lambda (sebelumnya dikenal sebagai otorisasi kustom) dari TOKEN jenisnya, Anda harus menentukan header kustom sebagai Sumber Token saat Anda mengonfigurasi otorisasi untuk API Anda. Klien API harus meneruskan token otorisasi yang diperlukan di header tersebut dalam permintaan yang masuk. Setelah menerima permintaan metode masuk, API Gateway mengekstrak token dari header kustom. Kemudian melewati token sebagai `authorizationToken` properti dari event objek fungsi Lambda, selain metode ARN sebagai properti: `methodArn`

```
{
  "type": "TOKEN",
  "authorizationToken": "{caller-supplied-token}",
  "methodArn": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}]/[{child-resources}]"
}
```

Dalam contoh ini, `type` properti menentukan jenis authorizer, yang merupakan TOKEN authorizer. `{caller-supplied-token}` Berasal dari header otorisasi dalam permintaan klien, dan dapat berupa nilai string apa pun. `methodArn` ini adalah ARN dari permintaan metode yang masuk dan diisi oleh API Gateway sesuai dengan konfigurasi otorisasi Lambda.

REQUEST format masukan

Untuk REQUEST jenis pengotorisasi Lambda, API Gateway meneruskan parameter permintaan ke fungsi Lambda otorisasi sebagai bagian dari objek. event Parameter permintaan termasuk header, parameter jalur, parameter string kueri, variabel tahap, dan beberapa variabel konteks permintaan. Pemanggil API dapat mengatur parameter jalur, header, dan parameter string kueri. Pengembang API harus menyetel variabel stage selama penerapan API dan API Gateway menyediakan konteks permintaan pada waktu berjalan.

 Note

Parameter jalur dapat diteruskan sebagai parameter permintaan ke fungsi otorisasi Lambda, tetapi tidak dapat digunakan sebagai sumber identitas.

Contoh berikut menunjukkan input ke REQUEST authorizer untuk metode API (GET /request) dengan integrasi proxy:

```
{
  "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"
}
}

```

requestContextIni adalah peta pasangan kunci-nilai dan sesuai dengan variabel [\\$context](#). Hasilnya bergantung pada API.

API Gateway mungkin menambahkan kunci baru ke peta. Untuk informasi selengkapnya tentang input fungsi Lambda dalam integrasi proxy Lambda, lihat. [Format input fungsi Lambda untuk integrasi proxy](#)

Keluaran dari otorisasi API Gateway Lambda

Output fungsi otorisasi Lambda adalah objek mirip kamus, yang harus menyertakan pengenalan utama (principalId) dan dokumen kebijakan () yang berisi daftar pernyataan kebijakan. policyDocument Outputnya juga dapat menyertakan context peta yang berisi pasangan kunci-nilai. Jika API menggunakan paket penggunaan ([apiKeySource](#) disetel ke AUTHORIZER), fungsi otorisasi Lambda harus mengembalikan salah satu kunci API paket penggunaan sebagai nilai properti. usageIdentifierKey

Berikut ini menunjukkan contoh output ini.

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}"
}

```

Di sini, pernyataan kebijakan menentukan apakah akan mengizinkan atau menolak (Effect) layanan eksekusi API Gateway untuk memanggil (Action) metode API yang ditentukan (Resource). Anda mungkin perlu mengontrol akses ke beberapa sumber daya berdasarkan otorisasi Anda. Anda dapat menggunakan wild card (*) untuk menentukan jenis sumber daya (metode). Untuk informasi tentang menyetel kebijakan yang valid untuk memanggil API, lihat [Referensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway](#).

Untuk metode ARN yang diaktifkan otorisasi, misalnya `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-resources}]]`, panjang maksimum adalah 1600 byte. Nilai parameter jalur, ukuran yang ditentukan pada waktu berjalan, dapat menyebabkan panjang ARN melebihi batas. Ketika ini terjadi, klien API akan menerima 414 Request URI too long respons.

Selain itu, ARN Sumber Daya, seperti yang ditunjukkan dalam keluaran pernyataan kebijakan oleh otorisasi, saat ini dibatasi hingga 512 karakter. Untuk alasan ini, Anda tidak boleh menggunakan URI dengan token JWT dengan panjang yang signifikan dalam URI permintaan. Anda dapat dengan aman meneruskan token JWT di header permintaan, sebagai gantinya.

Anda dapat mengakses `principalId` nilai dalam template pemetaan menggunakan `$context.authorizer.principalId` variabel. Ini berguna jika Anda ingin meneruskan nilai ke backend. Untuk informasi selengkapnya, lihat [Variabel konteks untuk transformasi data](#).

Anda dapat mengakses `stringKey`, `numberKey`, atau `booleanKey` nilai (misalnya, `"value"`, `"1"`, atau `"true"`) `context` peta dalam templat

pemetaan dengan memanggil `$context.authorizer.stringKey`, atau `$context.authorizer.numberKey` `$context.authorizer.booleanKey`, masing-masing. Nilai yang dikembalikan semuanya dirangkai. Perhatikan bahwa Anda tidak dapat mengatur objek atau array JSON sebagai nilai yang valid dari kunci apa pun di `context` peta.

Anda dapat menggunakan `context` peta untuk mengembalikan kredensi cache dari otorisasi ke backend, menggunakan templat pemetaan permintaan integrasi. Hal ini memungkinkan backend untuk memberikan pengalaman pengguna yang lebih baik dengan menggunakan kredensi cache untuk mengurangi kebutuhan untuk mengakses kunci rahasia dan membuka token otorisasi untuk setiap permintaan.

Untuk integrasi proxy Lambda, API Gateway meneruskan `context` objek dari otorisasi Lambda langsung ke fungsi Lambda backend sebagai bagian dari input. `event` Anda dapat mengambil pasangan `context` kunci-nilai dalam fungsi Lambda dengan memanggil `$event.requestContext.authorizer.key`

`{api-key}` singkatan dari kunci API dalam rencana penggunaan tahap API. Untuk informasi selengkapnya, lihat [the section called “Paket penggunaan”](#).

Berikut ini menunjukkan contoh output dari contoh Lambda authorizer. Output contoh berisi pernyataan kebijakan untuk memblokir (Deny) panggilan ke GET metode untuk dev tahap API (`ymy8tbxw7b`) dari AWS account (`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/"
      }
    ]
  }
}
```

Panggil API dengan otorisasi API Gateway Lambda

Setelah mengonfigurasi otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus) dan menerapkan API, Anda harus menguji API dengan otorisasi Lambda diaktifkan. [Untuk ini, Anda memerlukan klien REST, seperti cURL atau Postman.](#) Untuk contoh berikut, kami menggunakan Postman.

Note

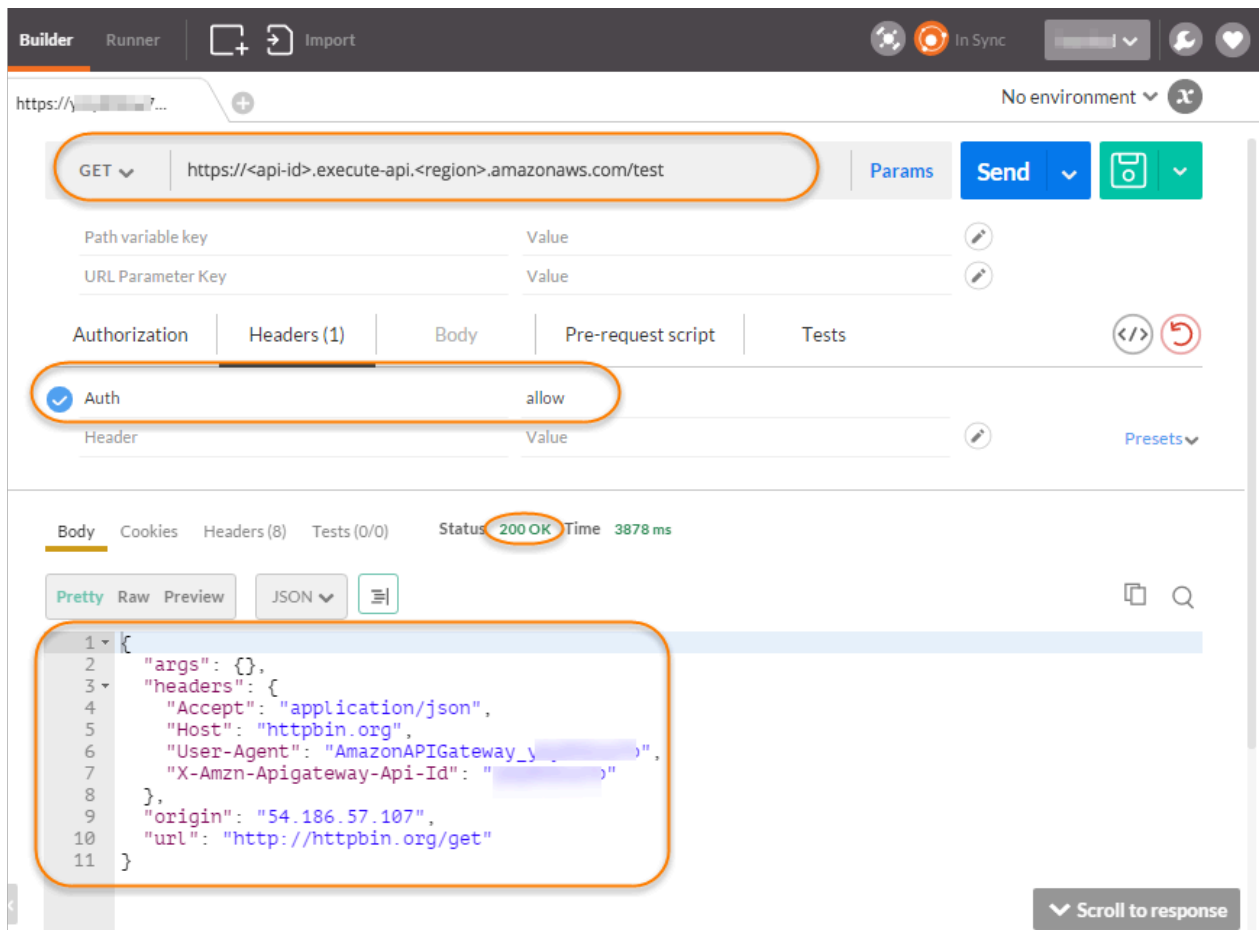
Saat memanggil metode berkemampuan pengotorisasi, API Gateway tidak mencatat panggilan CloudWatch jika token yang diperlukan untuk **TOKEN** otorisasi tidak disetel, nol, atau tidak valid oleh ekspresi validasi Token yang ditentukan. Demikian pula, API Gateway tidak mencatat panggilan CloudWatch jika salah satu sumber identitas yang diperlukan untuk REQUEST otorisasi tidak disetel, nol, atau kosong.

Berikut ini, kami menunjukkan cara menggunakan Postman untuk memanggil atau menguji API dengan otorisasi TOKEN Lambda. Metode ini dapat diterapkan untuk memanggil API dengan REQUEST otorisasi Lambda, jika Anda menentukan parameter path, header, atau string kueri yang diperlukan secara eksplisit.

Untuk memanggil API dengan **TOKEN** otorisasi khusus

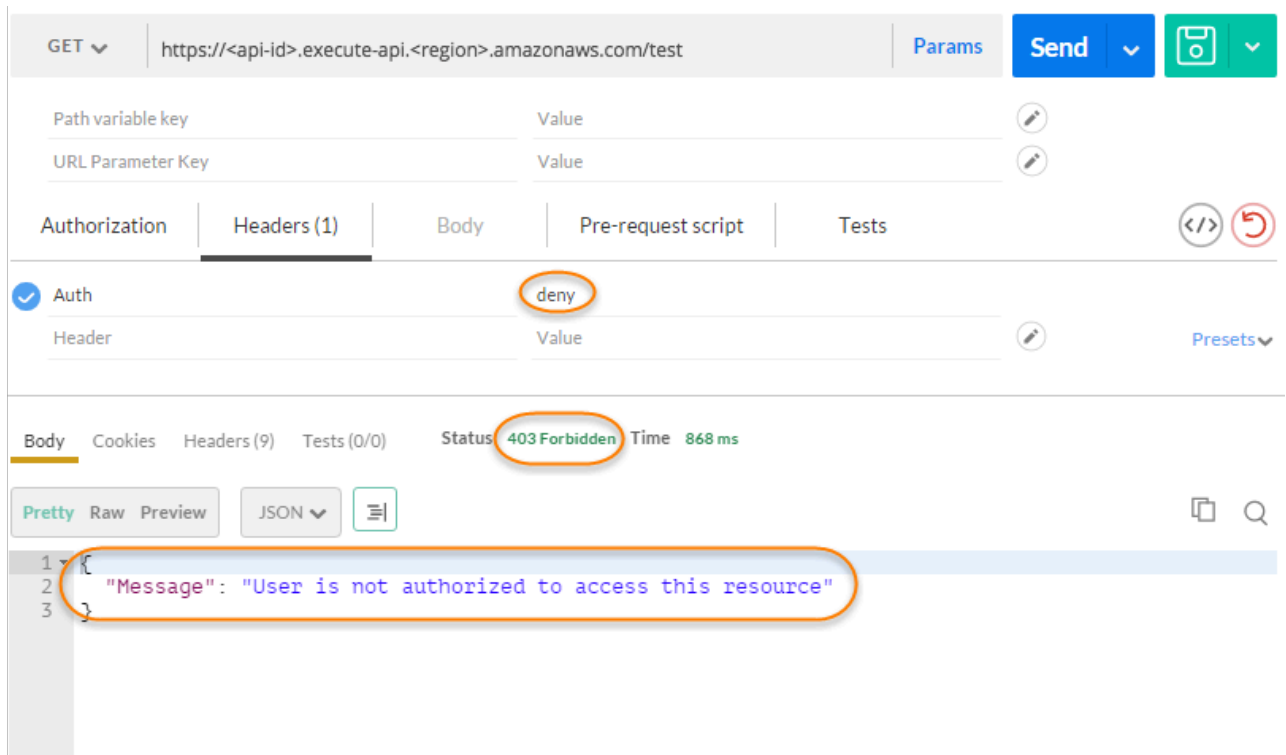
1. Buka Postman, pilih metode GET, dan tempel URL Invoke API ke bidang URL yang berdekatan.

Tambahkan header token otorisasi Lambda dan atur nilainya. `allow` Pilih Kirim.



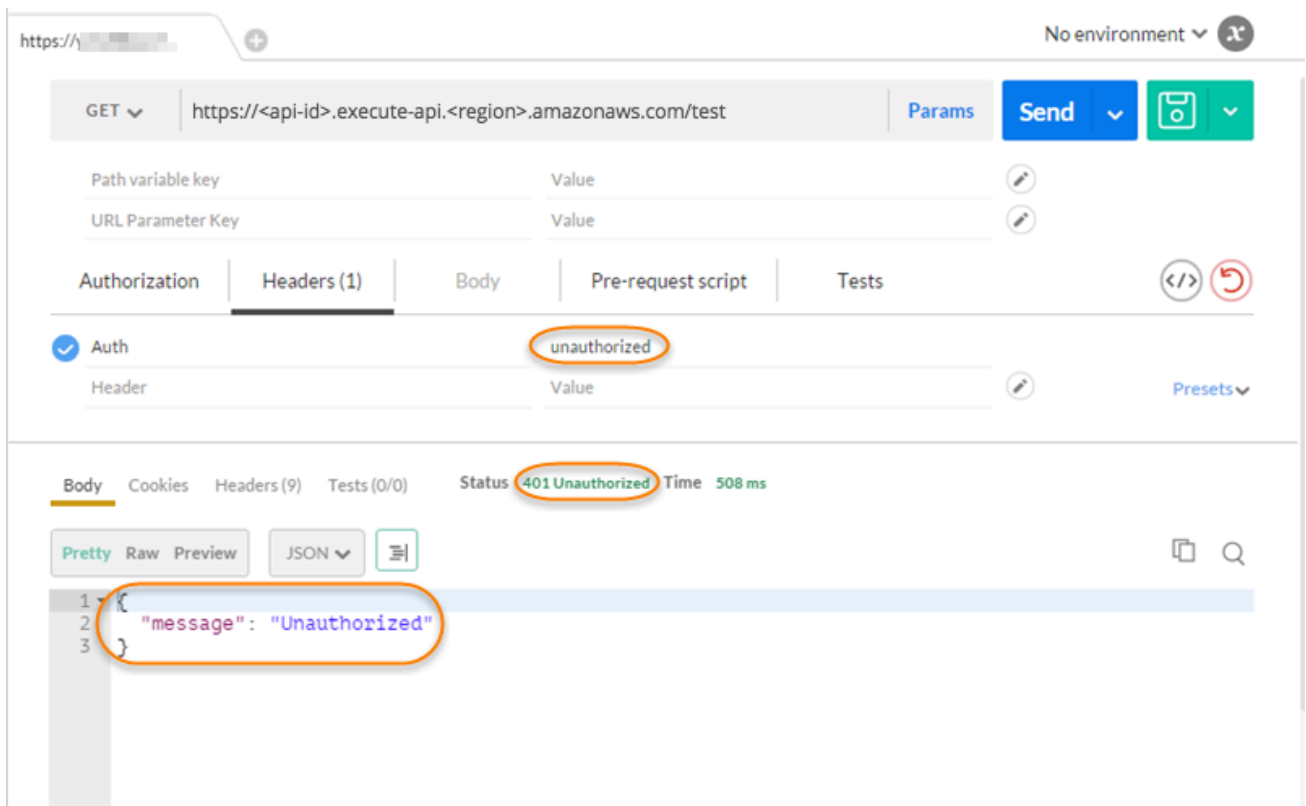
Tanggapan menunjukkan bahwa otorisasi API Gateway Lambda mengembalikan respons 200 OK dan berhasil mengotorisasi panggilan untuk mengakses titik akhir HTTP (`http://httpbin.org/get`) yang terintegrasi dengan metode.

2. Masih di Postman, ubah nilai header token otorisasi Lambda menjadi. deny Pilih Kirim.



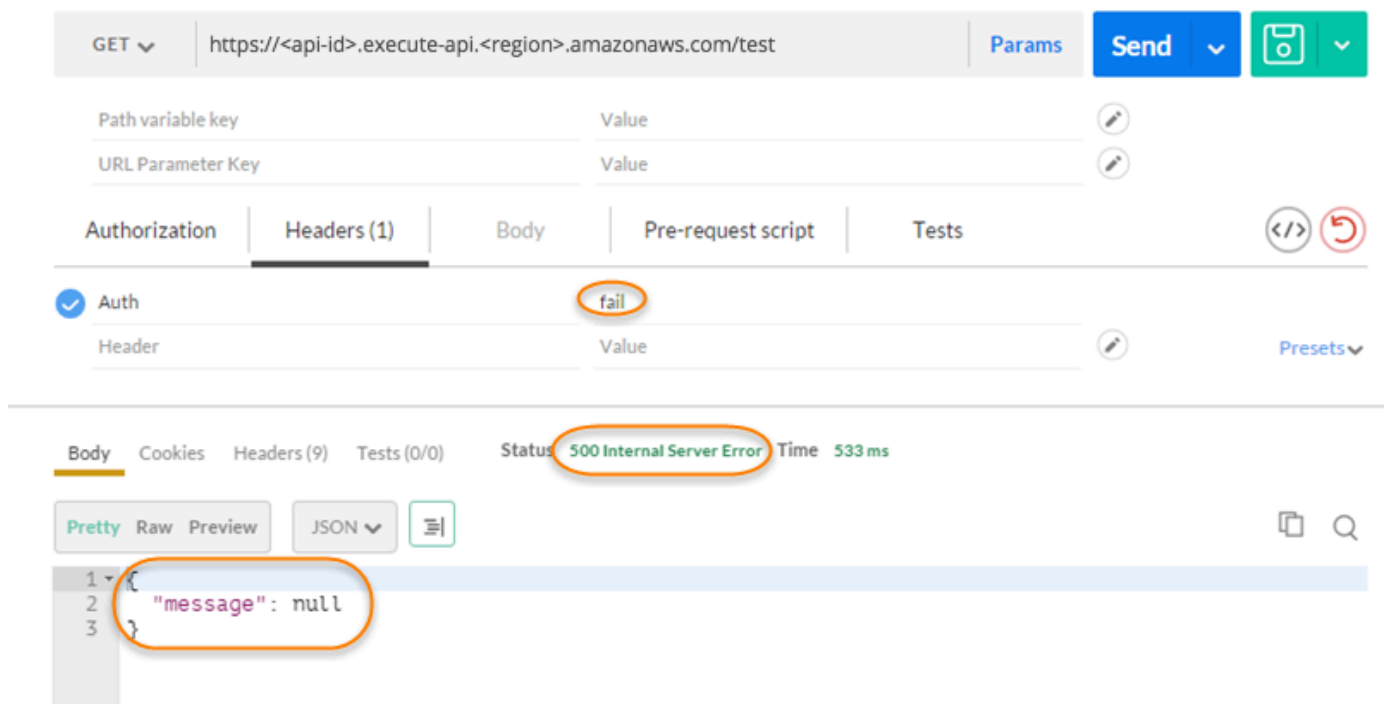
Tanggapan menunjukkan bahwa otorisasi API Gateway Lambda mengembalikan respons 403 Forbidden tanpa mengotorisasi panggilan untuk mengakses titik akhir HTTP.

3. Di Postman, ubah nilai **unauthorized** header token otorisasi Lambda menjadi dan pilih Kirim.



Respons menunjukkan bahwa API Gateway mengembalikan respons 401 Tidak Sah tanpa mengotorisasi panggilan untuk mengakses titik akhir HTTP.

4. Sekarang, ubah nilai header token otorisasi Lambda menjadi `fail`. Pilih Kirim.



Respons menunjukkan bahwa API Gateway mengembalikan respon 500 Internal Server Error tanpa mengotorisasi panggilan untuk mengakses endpoint HTTP.

Konfigurasi otorisasi Lambda API Gateway lintas akun

Anda sekarang juga dapat menggunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai fungsi otorisasi API Anda. Setiap akun dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Fungsi otorisasi Lambda dapat menggunakan strategi otentikasi token pembawa seperti atau SALL. OAuth Ini memudahkan untuk mengelola dan berbagi fungsi otorisasi Lambda pusat secara terpusat di beberapa API Gateway. APIs

Di bagian ini, kami menunjukkan cara mengonfigurasi fungsi otorisasi Lambda lintas akun menggunakan konsol Amazon API Gateway.

Petunjuk ini mengasumsikan bahwa Anda sudah memiliki API Gateway API di satu AWS akun dan fungsi otorisasi Lambda di akun lain.

Konfigurasi otorisasi Lambda lintas akun menggunakan konsol API Gateway

Masuk ke konsol Amazon API Gateway di akun yang memiliki API Anda di dalamnya, lalu lakukan hal berikut:

1. Pilih API Anda, lalu di panel navigasi utama, pilih Authorizers.
2. Pilih Buat Authorizer.
3. Untuk nama Authorizer, masukkan nama untuk otorisasi.
4. Untuk jenis Authorizer, pilih Lambda.
5. Untuk Fungsi Lambda, masukkan ARN lengkap untuk fungsi otorisasi Lambda yang Anda miliki di akun kedua Anda.

Note

Di konsol Lambda, Anda dapat menemukan ARN untuk fungsi Anda di sudut kanan atas jendela konsol.

6. Peringatan dengan string `aws lambda add-permission` perintah akan muncul. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi Lambda otorisasi. Salin perintah dan simpan untuk nanti. Anda menjalankan perintah setelah Anda membuat authorizer.

7. Untuk muatan acara Lambda, pilih Token untuk otorisasi atau Permintaan **TOKEN** otorisasi. REQUEST
8. Bergantung pada pilihan langkah sebelumnya, lakukan salah satu hal berikut:
 - a. Untuk opsi Token, lakukan hal berikut:
 - Untuk sumber Token, masukkan nama header yang berisi token otorisasi. Klien API harus menyertakan header dari nama ini untuk mengirim token otorisasi ke otorisasi Lambda.
 - Secara opsional, untuk validasi Token, masukkan pernyataan RegEx . API Gateway melakukan validasi awal token input terhadap ekspresi ini dan memanggil authorizer setelah validasi berhasil. Ini membantu mengurangi panggilan ke API Anda.
 - Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan. Saat caching kebijakan diaktifkan, nama header yang ditentukan dalam sumber Token menjadi kunci cache. Jika beberapa nilai diteruskan ke header ini dalam permintaan, semua nilai akan menjadi kunci cache, dengan urutan dipertahankan.

 Note

Nilai TTL default adalah 300 detik. Nilai maksimum adalah 3600 detik, batas ini tidak dapat ditingkatkan.

- b. Untuk opsi Permintaan, lakukan hal berikut:
 - Untuk tipe sumber Identity, pilih tipe parameter. Jenis parameter yang didukung adalah `Header`, `Query string`, `Stage variable`, dan `Context`. Untuk menambahkan lebih banyak sumber identitas, pilih Tambah parameter.
 - Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan.

API Gateway menggunakan sumber identitas yang ditentukan sebagai kunci caching otorisasi permintaan. Saat caching diaktifkan, API Gateway memanggil fungsi Lambda otorisasi hanya setelah berhasil memverifikasi bahwa semua sumber identitas yang ditentukan ada saat runtime. Jika sumber identitas tertentu hilang, null, atau kosong, API

Gateway mengembalikan 401 Unauthorized respons tanpa memanggil fungsi Lambda otorisasi.

Ketika beberapa sumber identitas didefinisikan, mereka semua digunakan untuk mendapatkan kunci cache otorisasi. Mengubah salah satu bagian kunci cache menyebabkan otorisasi membuang dokumen kebijakan yang di-cache dan menghasilkan yang baru. Jika header dengan beberapa nilai diteruskan dalam permintaan, maka semua nilai akan menjadi bagian dari kunci cache, dengan urutan dipertahankan.

- Ketika caching dimatikan, tidak perlu menentukan sumber identitas.

Note

Untuk mengaktifkan caching, otorisasi Anda harus mengembalikan kebijakan yang berlaku untuk semua metode di seluruh API. Untuk menerapkan kebijakan khusus metode, Anda dapat menonaktifkan caching Otorisasi.

9. Pilih Buat Authorizer.
10. Tempel string `aws lambda add-permission` perintah yang Anda salin pada langkah sebelumnya ke AWS CLI jendela yang dikonfigurasi untuk akun kedua Anda. Ganti `AUTHORIZER_ID` dengan ID otorisasi Anda. Ini akan memberikan akses akun pertama Anda ke fungsi otorisasi Lambda akun kedua Anda.

Kontrol akses berdasarkan atribut identitas dengan Izin Terverifikasi

Gunakan Izin Terverifikasi Amazon untuk mengontrol akses ke API Gateway API Anda. Saat Anda menggunakan API Gateway dengan Izin Terverifikasi, Izin Terverifikasi akan membuat otorisasi Lambda yang menggunakan keputusan otorisasi berbutir halus untuk mengontrol akses ke API Anda. Izin Terverifikasi mengotorisasi pemanggil berdasarkan skema penyimpanan kebijakan dan kebijakan menggunakan bahasa kebijakan Cedar untuk menentukan izin berbutir halus bagi pengguna aplikasi. Untuk informasi selengkapnya, lihat [Membuat penyimpanan kebijakan dengan API dan penyedia identitas yang terhubung](#) di Panduan Pengguna Izin Terverifikasi Amazon.

Izin Terverifikasi mendukung kumpulan pengguna Amazon Cognito atau penyedia identitas OpenID Connect (OIDC) sebagai sumber identitas. Izin Terverifikasi menganggap bahwa prinsipal telah diidentifikasi dan diautentikasi sebelumnya. Izin Terverifikasi hanya didukung untuk REST Regional dan yang dioptimalkan di tepi. APIs

Buat otorisasi Lambda menggunakan Izin Terverifikasi

Izin Terverifikasi membuat otorisasi Lambda untuk menentukan apakah prinsipal diizinkan melakukan tindakan pada API Anda. Anda membuat kebijakan Cedar yang digunakan Izin Terverifikasi untuk melakukan tugas otorisasi.

Berikut ini adalah contoh kebijakan Cedar yang memungkinkan akses untuk menjalankan API berdasarkan kumpulan pengguna Amazon Cognito-`us-east-1_ABC1234`, untuk grup `developer` pada `GET /users` sumber daya API. Izin Terverifikasi menentukan keanggotaan grup dengan mengurai token pembawa untuk identitas pemanggil.

```
permit(  
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",  
  action in [ MyAPI::Action::"get /users" ],  
  resource  
);
```

Secara opsional, Izin Terverifikasi dapat melampirkan otorisasi ke metode API Anda. Pada tahap produksi untuk API Anda, kami menyarankan agar Anda tidak mengizinkan Izin Terverifikasi untuk melampirkan otorisasi untuk Anda.

Daftar berikut menunjukkan cara mengonfigurasi Izin Terverifikasi untuk melampirkan atau tidak melampirkan otorisasi Lambda ke permintaan metode metode API Anda.

Lampirkan otorisasi untuk Anda ()Konsol Manajemen AWS

Saat Anda memilih Buat penyimpanan kebijakan di konsol Izin Terverifikasi, pada halaman integrasi aplikasi Deploy, pilih Sekarang.

Jangan lampirkan otorisasi untuk Anda ()Konsol Manajemen AWS

Saat Anda memilih Buat penyimpanan kebijakan di konsol Izin Terverifikasi, pada halaman Integrasi aplikasi Terapkan, pilih Nanti.

Izin Terverifikasi masih membuat otorisasi Lambda untuk Anda. Otorisasi Lambda dimulai dengan `AVPAuthorizerLambda`- Untuk petunjuk selengkapnya tentang cara melampirkan otorisasi Anda pada suatu metode, lihat [the section called "Konfigurasi metode untuk menggunakan otorisasi Lambda \(konsol\)"](#).

Lampirkan otorisasi untuk Anda ()CloudFormation

Di CloudFormation template yang dihasilkan Izin Terverifikasi, di `Conditions` bagian, atur ke `"Ref": "shouldAttachAuthorizer" true`

Jangan lampirkan otorisasi untuk Anda ()CloudFormation

Di CloudFormation template yang dihasilkan Izin Terverifikasi, di Conditions bagian, atur ke.
"Ref": "shouldAttachAuthorizer" false

Izin Terverifikasi masih membuat otorisasi Lambda untuk Anda. Otorisasi Lambda dimulai dengan. AVPAuthorizerLambda- Untuk petunjuk selengkapnya tentang cara melampirkan otorisasi Anda pada suatu metode, lihat [the section called “Konfigurasi metode untuk menggunakan otorisasi Lambda \(\)AWS CLI”](#).

Panggil otorisasi Lambda menggunakan Izin Terverifikasi

Anda dapat menghubungi otorisasi Lambda Anda dengan memberikan identitas atau token akses di header. Authorization Untuk informasi selengkapnya, lihat [the section called “Panggil API dengan otorisasi Lambda”](#).

API Gateway menyimpan kebijakan yang dikembalikan oleh otorisasi Lambda Anda selama 120 detik. Anda dapat memodifikasi TTL di konsol API Gateway atau dengan menggunakan. AWS CLI

Kontrol akses ke REST APIs menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi

Sebagai alternatif untuk menggunakan [peran dan kebijakan IAM](#) atau otorisasi [Lambda](#) (sebelumnya dikenal sebagai otorisasi khusus), Anda dapat menggunakan kumpulan [pengguna Amazon Cognito untuk mengontrol siapa yang dapat mengakses API Anda di Amazon](#) API Gateway.

Untuk menggunakan kumpulan pengguna Amazon Cognito dengan API Anda, Anda harus terlebih dahulu membuat otorisasi COGNITO_USER_POOLS jenis tersebut dan kemudian mengonfigurasi metode API untuk menggunakan otorisasi tersebut. Setelah API diterapkan, klien harus terlebih dahulu memasukkan pengguna ke kumpulan pengguna, mendapatkan [identitas atau token akses](#) untuk pengguna, dan kemudian memanggil metode API dengan salah satu token, yang biasanya diatur ke Authorization header permintaan. Panggilan API hanya berhasil jika token yang diperlukan disediakan dan token yang disediakan valid, jika tidak, klien tidak berwenang untuk melakukan panggilan karena klien tidak memiliki kredensi yang dapat diotorisasi.

Token identitas digunakan untuk mengotorisasi panggilan API berdasarkan klaim identitas pengguna yang masuk. Token akses digunakan untuk mengotorisasi panggilan API berdasarkan cakupan kustom sumber daya yang dilindungi akses tertentu. Untuk informasi selengkapnya, lihat [Menggunakan Token dengan Kumpulan Pengguna](#) dan [Server Sumber Daya dan Cakupan Khusus](#).

Untuk membuat dan mengonfigurasi kumpulan pengguna Amazon Cognito untuk API, Anda melakukan tugas berikut:

- Gunakan konsol Amazon Cognito, CLI/SDK, atau API untuk membuat kumpulan pengguna—atau gunakan yang dimiliki oleh akun lain. AWS
- Gunakan konsol API Gateway, CLI/SDK, atau API untuk membuat otorisasi API Gateway dengan kumpulan pengguna yang dipilih.
- Gunakan konsol API Gateway, CLI/SDK, atau API untuk mengaktifkan otorisasi pada metode API yang dipilih.

Untuk memanggil metode API apa pun dengan kumpulan pengguna diaktifkan, klien API Anda melakukan tugas berikut:

- Gunakan Amazon Cognito CLI/SDK atau API untuk menandatangani pengguna ke kumpulan pengguna yang dipilih, dan mendapatkan token identitas atau token akses. Untuk mempelajari lebih lanjut tentang menggunakan SDKs, lihat [Contoh kode untuk Amazon Cognito](#) menggunakan AWS SDKs
- Gunakan kerangka kerja khusus klien untuk memanggil API Gateway API yang diterapkan dan menyediakan token yang sesuai di header. Authorization

Sebagai pengembang API, Anda harus memberikan ID kumpulan pengguna, ID klien, dan mungkin rahasia klien terkait yang didefinisikan sebagai bagian dari kumpulan pengguna.

 Note

[Untuk mengizinkan pengguna masuk menggunakan kredensi Amazon Cognito dan juga mendapatkan kredensi sementara untuk digunakan dengan izin peran IAM, gunakan Identitas Federasi Amazon Cognito.](#) Untuk setiap metode HTTP titik akhir sumber daya API, tetapkan jenis otorisasi, kategoriMethod Execution, ke. AWS_IAM

Di bagian ini, kami menjelaskan cara membuat kumpulan pengguna, cara mengintegrasikan API Gateway API dengan kumpulan pengguna, dan cara menjalankan API yang terintegrasi dengan kumpulan pengguna.

Topik

- [Buat kumpulan pengguna Amazon Cognito untuk REST API](#)

- [Integrasikan REST API dengan kumpulan pengguna Amazon Cognito](#)
- [Panggil REST API yang terintegrasi dengan kumpulan pengguna Amazon Cognito](#)
- [Konfigurasi otorisasi Amazon Cognito lintas akun untuk REST API menggunakan konsol API Gateway](#)
- [Buat otorisasi Amazon Cognito untuk REST API menggunakan CloudFormation](#)

Buat kumpulan pengguna Amazon Cognito untuk REST API

Sebelum mengintegrasikan API Anda dengan kumpulan pengguna, Anda harus membuat kumpulan pengguna di Amazon Cognito. Konfigurasi kumpulan pengguna Anda harus mengikuti semua [kuota sumber daya untuk Amazon Cognito](#). Semua variabel Amazon Cognito yang ditentukan pengguna seperti grup, pengguna, dan peran hanya boleh menggunakan karakter alfanumerik. Untuk petunjuk tentang cara membuat kumpulan pengguna, lihat [Tutorial: Membuat kumpulan pengguna](#) di Panduan Pengembang Amazon Cognito.

Perhatikan ID kumpulan pengguna, ID klien, dan rahasia klien apa pun. Klien harus menyediakannya ke Amazon Cognito agar pengguna dapat mendaftar dengan kumpulan pengguna, masuk ke kumpulan pengguna, dan untuk mendapatkan identitas atau token akses untuk disertakan dalam permintaan memanggil metode API yang dikonfigurasi dengan kumpulan pengguna. Selain itu, Anda harus menentukan nama kumpulan pengguna saat mengonfigurasi kumpulan pengguna sebagai otorisasi di API Gateway, seperti yang dijelaskan selanjutnya.

Jika Anda menggunakan token akses untuk mengotorisasi panggilan metode API, pastikan untuk mengonfigurasi integrasi aplikasi dengan kumpulan pengguna untuk menyiapkan cakupan kustom yang Anda inginkan di server sumber daya tertentu. Untuk informasi selengkapnya tentang penggunaan token dengan kumpulan pengguna Amazon Cognito, lihat [Menggunakan Token dengan Kumpulan Pengguna](#). Untuk informasi selengkapnya tentang server sumber daya, lihat [Mendefinisikan Server Sumber Daya untuk Kumpulan Pengguna Anda](#).

Perhatikan pengidentifikasi server sumber daya yang dikonfigurasi dan nama cakupan kustom. Anda membutuhkannya untuk membangun nama lengkap cakupan akses untuk OAuth Scopes, yang digunakan oleh otorisasi. COGNITO_USER_POOLS

The screenshot shows the Amazon Cognito console interface. On the left is a navigation menu with categories: User management (Users, Groups), Authentication (Authentication methods, Sign-in, Sign-up, Social and external providers, Extensions), Security (AWS WAF, Threat protection, Log streaming), and Branding (Domain, Managed login, Message templates, Settings, Identity pools). The 'Domain' page is selected. It features a 'Domain' section with an 'Info' icon and an 'Actions' dropdown. Below this are two columns: 'Cognito domain' and 'Custom domain', each showing 'Domain' and 'Branding version' as '-'. The 'Resource servers (1)' section has an 'Info' icon and buttons for 'Edit', 'Delete', and 'Create resource server'. A search bar is present. Below the search bar is a table with columns: 'Resource server name', 'Resource server identifier', and 'Custom scopes'. The table contains one entry: 'PetStore' with identifier 'https://my-petstore-api.example.com' and '2 custom scopes'. The identifier and custom scopes are highlighted with red boxes in the original image.

Integrasikan REST API dengan kumpulan pengguna Amazon Cognito

Setelah membuat kumpulan pengguna Amazon Cognito, di API Gateway, Anda harus membuat COGNITO_USER_POOLS otorisasi yang menggunakan kumpulan pengguna. Prosedur berikut menunjukkan cara melakukannya menggunakan konsol API Gateway.

Note

Anda dapat menggunakan [CreateAuthorizer](#) tindakan untuk membuat COGNITO_USER_POOLS otorisasi yang menggunakan beberapa kumpulan pengguna. Anda dapat menggunakan hingga 1.000 kumpulan pengguna untuk satu COGNITO_USER_POOLS otorisasi. Batas ini tidak dapat dinaikkan.

Important

Setelah melakukan salah satu prosedur di bawah ini, Anda harus menerapkan atau menerapkan ulang API Anda untuk menyebarkan perubahan. Untuk informasi selengkapnya tentang penerapan API Anda, lihat [Menerapkan REST APIs di API Gateway](#).

Untuk membuat **COGNITO_USER_POOLS** otorisasi menggunakan konsol API Gateway

1. Buat API baru, atau pilih API yang ada di API Gateway.
2. Di panel navigasi utama, pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk mengonfigurasi otorisasi baru untuk menggunakan kumpulan pengguna, lakukan hal berikut:
 - a. Untuk nama Authorizer, masukkan nama.
 - b. Untuk jenis Authorizer, pilih Cognito.
 - c. Untuk kumpulan pengguna Cognito, pilih Wilayah AWS tempat Anda membuat Amazon Cognito dan pilih kumpulan pengguna yang tersedia.

Anda dapat menggunakan variabel tahap untuk menentukan kumpulan pengguna Anda. Gunakan format berikut untuk kumpulan pengguna Anda: `arn:aws:cognito-idp:us-east-2:111122223333:userpool/${stageVariables.MyUserPool}`.

- d. Untuk sumber Token, masukkan **Authorization** sebagai nama header untuk meneruskan identitas atau token akses yang dikembalikan oleh Amazon Cognito saat pengguna berhasil masuk.
 - e. (Opsional) Masukkan ekspresi reguler di bidang validasi Token untuk memvalidasi bidang aud (audiens) token identitas sebelum permintaan diotorisasi dengan Amazon Cognito. Perhatikan bahwa saat menggunakan token akses validasi ini menolak permintaan karena token akses tidak berisi bidang. aud
 - f. Pilih Buat Authorizer.
5. Setelah membuat **COGNITO_USER_POOLS** otorisasi, Anda dapat menguji panggilannya dengan menyediakan token identitas yang disediakan dari kumpulan pengguna. Anda tidak dapat menggunakan token akses untuk menguji pemanggilan otorisasi Anda.

Anda dapat memperoleh token identitas ini dengan memanggil [Amazon Cognito Identity SDK](#) untuk melakukan login pengguna. Anda juga dapat menggunakan [InitiateAuth](#) aksinya. Jika Anda tidak mengonfigurasi cakupan Otorisasi apa pun, API Gateway memperlakukan token yang disediakan sebagai token identitas.

Prosedur sebelumnya membuat **COGNITO_USER_POOLS** otorisasi yang menggunakan kumpulan pengguna Amazon Cognito yang baru dibuat. Bergantung pada cara Anda mengaktifkan otorisasi

pada metode API, Anda dapat menggunakan token identitas atau token akses yang disediakan dari kumpulan pengguna terintegrasi.

Untuk mengkonfigurasi **COGNITO_USER_POOLS** otorisasi pada metode

1. Pilih Sumber daya. Pilih metode baru atau pilih metode yang ada. Jika perlu, buat sumber daya.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk Authorizer, dari menu tarik-turun, pilih otorisasi kumpulan pengguna Amazon Cognito yang baru saja Anda buat.
4. Untuk menggunakan token identitas, lakukan hal berikut:
 - a. Jaga Cakupan Otorisasi kosong.
 - b. Jika diperlukan, dalam permintaan Integrasi, tambahkan `$context.authorizer.claims.property-name` ekspresi `$context.authorizer.claims['property-name']` atau dalam templat pemetaan tubuh untuk meneruskan properti klaim identitas yang ditentukan dari kumpulan pengguna ke backend. Untuk nama properti sederhana, seperti `sub` atau `custom-sub`, dua notasi identik. Untuk nama properti kompleks, seperti `custom:role`, Anda tidak dapat menggunakan notasi titik. Misalnya, ekspresi pemetaan berikut meneruskan [bidang standar](#) klaim `sub` dan `email` ke backend:

```
{
  "context" : {
    "sub" : "$context.authorizer.claims.sub",
    "email" : "$context.authorizer.claims.email"
  }
}
```

Jika Anda mendeklarasikan bidang klaim kustom saat mengonfigurasi kumpulan pengguna, Anda dapat mengikuti pola yang sama untuk mengakses bidang kustom. Contoh berikut mendapatkan `role` bidang khusus klaim:

```
{
  "context" : {
    "role" : "$context.authorizer.claims.role"
  }
}
```


Jika bidang klaim kustom dideklarasikan sebagaicustom:role, gunakan contoh berikut untuk mendapatkan properti klaim:

```
{
  "context" : {
    "role" : "$context.authorizer.claims['custom:role']"
  }
}
```

5. Untuk menggunakan token akses, lakukan hal berikut:
 - a. Untuk Cakupan Otorisasi, masukkan satu atau beberapa nama lengkap cakupan yang telah dikonfigurasi saat kumpulan pengguna Amazon Cognito dibuat. Misalnya, mengikuti contoh yang diberikan[Buat kumpulan pengguna Amazon Cognito untuk REST API](#), salah satu cakupannya adalahhttps://my-petstore-api.example.com/cats.read.

Saat runtime, pemanggilan metode berhasil jika cakupan apa pun yang ditentukan pada metode dalam langkah ini cocok dengan cakupan yang diklaim dalam token yang masuk. Jika tidak, panggilan gagal dengan 401 Unauthorized respons.
 - b. Pilih Simpan.
6. Ulangi langkah-langkah ini untuk metode lain yang Anda pilih.

Dengan COGNITO_USER_POOLS otorisasi, jika opsi OAuthScopes tidak ditentukan, API Gateway memperlakukan token yang disediakan sebagai token identitas dan memverifikasi identitas yang diklaim terhadap yang dari kumpulan pengguna. Jika tidak, API Gateway memperlakukan token yang disediakan sebagai token akses dan memverifikasi cakupan akses yang diklaim dalam token terhadap cakupan otorisasi yang dideklarasikan pada metode.

Alih-alih menggunakan konsol API Gateway, Anda juga dapat mengaktifkan kumpulan pengguna Amazon Cognito pada metode dengan menentukan file definisi OpenAPI dan mengimpor definisi API ke API Gateway.

Untuk mengimpor otorisasi COGNITO_USER_POOLS dengan file definisi OpenAPI

1. Buat (atau ekspor) file definisi OpenAPI untuk API Anda.
2. Tentukan definisi JSON COGNITO_USER_POOLS authorizer (MyUserPool) sebagai bagian dari securitySchemes bagian di OpenAPI 3.0 atau securityDefinitions bagian di Open API 2.0 sebagai berikut:

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. Untuk menggunakan token identitas untuk otorisasi metode, tambahkan { "MyUserPool": [] } ke security definisi metode, seperti yang ditunjukkan dalam metode GET berikut pada sumber daya root.

```
"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. Untuk menggunakan token akses untuk otorisasi metode, ubah definisi keamanan di atas menjadi `{ "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"
  }
},
...
}
```

5. Jika diperlukan, Anda dapat mengatur pengaturan konfigurasi API lainnya dengan menggunakan definisi atau ekstensi OpenAPI yang sesuai. Untuk informasi selengkapnya, lihat [Ekstensi OpenAPI untuk API Gateway](#).

Panggil REST API yang terintegrasi dengan kumpulan pengguna Amazon Cognito

Untuk memanggil metode dengan otorisasi kumpulan pengguna yang dikonfigurasi, klien harus melakukan hal berikut:

- Aktifkan pengguna untuk mendaftar dengan kumpulan pengguna.
- Aktifkan pengguna untuk masuk ke kumpulan pengguna.
- Dapatkan [identitas atau token akses](#) pengguna yang masuk dari kumpulan pengguna.
- Sertakan token di Authorization header (atau header lain yang Anda tentukan saat Anda membuat otorisasi).

Anda dapat menggunakan [AWS Amplify](#) untuk melakukan tugas-tugas ini. Lihat [Mengintegrasikan Amazon Cognito Dengan Web dan Aplikasi Seluler](#) untuk informasi selengkapnya.

- Untuk Android, lihat [Memulai dengan Amplify untuk Android](#).
- Untuk menggunakan iOS, lihat [Memulai dengan Amplify untuk iOS](#).
- Untuk menggunakannya JavaScript, lihat [Memulai dengan Amplify for Javascript](#).

Konfigurasi otorisasi Amazon Cognito lintas akun untuk REST API menggunakan konsol API Gateway

Anda sekarang juga dapat menggunakan kumpulan pengguna Amazon Cognito dari AWS akun lain sebagai otorisasi API Anda. Kumpulan pengguna Amazon Cognito dapat menggunakan strategi otentikasi token pembawa seperti atau SALL. OAuth Hal ini memudahkan untuk mengelola dan berbagi otorisasi kumpulan pengguna Amazon Cognito pusat di beberapa API Gateway. APIs

Di bagian ini, kami menunjukkan cara mengonfigurasi kumpulan pengguna Amazon Cognito lintas akun menggunakan konsol Amazon API Gateway.

Petunjuk ini mengasumsikan bahwa Anda sudah memiliki API Gateway API di satu AWS akun dan kumpulan pengguna Amazon Cognito di akun lain.

Buat otorisasi Amazon Cognito lintas akun untuk REST API

Masuk ke konsol Amazon API Gateway di akun yang memiliki API Anda di dalamnya, lalu lakukan hal berikut:

1. Buat API baru, atau pilih API yang ada di API Gateway.
2. Di panel navigasi utama, pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk mengonfigurasi otorisasi baru untuk menggunakan kumpulan pengguna, lakukan hal berikut:
 - a. Untuk nama Authorizer, masukkan nama.
 - b. Untuk jenis Authorizer, pilih Cognito.
 - c. Untuk kumpulan pengguna Cognito, masukkan ARN lengkap untuk kumpulan pengguna yang Anda miliki di akun kedua Anda.

Note

Di konsol Amazon Cognito, Anda dapat menemukan ARN untuk kumpulan pengguna Anda di bidang ARN Kolam pada panel Pengaturan Umum.

- d. Untuk sumber Token, masukkan **Authorization** sebagai nama header untuk meneruskan identitas atau token akses yang dikembalikan oleh Amazon Cognito saat pengguna berhasil masuk.
- e. (Opsional) Masukkan ekspresi reguler di bidang validasi Token untuk memvalidasi bidang aud (audiens) token identitas sebelum permintaan diotorisasi dengan Amazon Cognito. Perhatikan bahwa saat menggunakan token akses validasi ini menolak permintaan karena token akses tidak berisi bidang. aud
- f. Pilih Buat Authorizer.

Buat otorisasi Amazon Cognito untuk REST API menggunakan CloudFormation

Anda dapat menggunakannya CloudFormation untuk membuat kumpulan pengguna Amazon Cognito dan otorisasi Amazon Cognito. Contoh CloudFormation template melakukan hal berikut:

- Buat kumpulan pengguna Amazon Cognito. Klien harus terlebih dahulu menandatangani pengguna ke kumpulan pengguna dan mendapatkan [identitas atau token akses](#). Jika Anda menggunakan

token akses untuk mengotorisasi panggilan metode API, pastikan untuk mengonfigurasi integrasi aplikasi dengan kumpulan pengguna untuk menyiapkan cakupan kustom yang Anda inginkan di server sumber daya tertentu.

- Membuat API Gateway API dengan GET metode.
- Membuat otorisasi Amazon Cognito yang menggunakan Authorization header sebagai sumber token.

```

AWS::ApiGateway::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
- /
```

Integrasi untuk REST APIs di API Gateway

Setelah menyiapkan metode API, Anda harus mengintegrasikannya dengan titik akhir di backend. Endpoint backend juga disebut sebagai titik akhir integrasi dan dapat berupa fungsi Lambda, halaman web HTTP, atau tindakan layanan. AWS

Seperti halnya metode API, integrasi API memiliki permintaan integrasi dan respons integrasi. Permintaan integrasi merangkum permintaan HTTP yang diterima oleh backend. Ini mungkin atau mungkin tidak berbeda dari permintaan metode yang diajukan oleh klien. Respons integrasi adalah respons HTTP yang merangkum output yang dikembalikan oleh backend.

Menyiapkan permintaan integrasi melibatkan hal-hal berikut: mengonfigurasi cara meneruskan permintaan metode yang dikirimkan klien ke backend; mengonfigurasi cara mengubah data permintaan, jika perlu, ke data permintaan integrasi; dan menentukan fungsi Lambda mana yang akan dipanggil, menentukan server HTTP mana yang akan meneruskan permintaan masuk, atau menentukan tindakan layanan yang akan dipanggil. AWS

Menyiapkan respons integrasi (hanya berlaku untuk integrasi non-proxy) melibatkan hal berikut: mengonfigurasi cara meneruskan hasil yang dikembalikan ke respons metode dari kode status tertentu, mengonfigurasi cara mengubah parameter respons integrasi yang ditentukan ke parameter respons metode yang telah dikonfigurasi sebelumnya, dan mengonfigurasi cara memetakan badan respons integrasi ke badan respons metode sesuai dengan templat pemetaan tubuh yang ditentukan.

Secara terprogram, permintaan integrasi dienkapsulasi oleh [Integration](#) sumber daya dan respons integrasi oleh sumber daya API Gateway. [IntegrationResponse](#)

Untuk menyiapkan permintaan integrasi, Anda membuat [Integration](#) sumber daya dan menggunakannya untuk mengonfigurasi URL titik akhir integrasi. Anda kemudian mengatur izin IAM untuk mengakses backend, dan menentukan pemetaan untuk mengubah data permintaan yang masuk sebelum meneruskannya ke backend. Untuk menyiapkan respons integrasi untuk integrasi

non-proxy, Anda membuat [IntegrationResponse](#) sumber daya dan menggunakannya untuk menetapkan respons metode targetnya. Anda kemudian mengonfigurasi cara memetakan output backend ke respons metode.

Topik

- [Menyiapkan permintaan integrasi di API Gateway](#)
- [Siapkan respons integrasi di API Gateway](#)
- [Integrasi Lambda untuk REST APIs di API Gateway](#)
- [Integrasi HTTP untuk REST APIs di API Gateway](#)
- [Streaming respons integrasi untuk integrasi proxy Anda di API Gateway](#)
- [Integrasi pribadi untuk REST APIs di API Gateway](#)
- [Integrasi tiruan untuk REST APIs di API Gateway](#)

Menyiapkan permintaan integrasi di API Gateway

Untuk menyiapkan permintaan integrasi, Anda melakukan tugas wajib dan opsional berikut:

1. Pilih jenis integrasi yang menentukan bagaimana data permintaan metode diteruskan ke backend.
2. Untuk integrasi non-tiruan, tentukan metode HTTP dan URI dari titik akhir integrasi yang ditargetkan, kecuali untuk integrasi. MOCK
3. Untuk integrasi dengan fungsi Lambda dan tindakan layanan AWS lainnya, tetapkan peran IAM dengan izin yang diperlukan untuk API Gateway untuk memanggil backend atas nama Anda.
4. Untuk integrasi non-proxy, atur pemetaan parameter yang diperlukan untuk memetakan parameter permintaan metode yang telah ditentukan sebelumnya ke parameter permintaan integrasi yang sesuai.
5. Untuk integrasi non-proxy, setel pemetaan badan yang diperlukan untuk memetakan badan permintaan metode masuk dari jenis konten tertentu sesuai dengan templat pemetaan yang ditentukan.
6. Untuk integrasi non-proxy, tentukan kondisi di mana data permintaan metode masuk diteruskan ke backend apa adanya.
7. Secara opsional, tentukan cara menangani konversi tipe untuk muatan biner.
8. Secara opsional, deklarasikan nama namespace cache dan parameter kunci cache untuk mengaktifkan caching API.

Melakukan tugas ini melibatkan pembuatan sumber daya [Integrasi](#) API Gateway dan menetapkan nilai properti yang sesuai. Anda dapat melakukannya menggunakan konsol API Gateway, AWS CLI perintah, AWS SDK, atau API Gateway REST API.

Topik

- [Tugas dasar permintaan integrasi API](#)
- [Pilih jenis integrasi API Gateway API](#)
- [Siapkan integrasi proxy dengan sumber daya proxy](#)
- [Menyiapkan permintaan integrasi API menggunakan konsol API Gateway](#)

Tugas dasar permintaan integrasi API

Permintaan integrasi adalah permintaan HTTP yang API Gateway kirimkan ke backend, meneruskan data permintaan yang dikirimkan klien, dan mengubah data, jika perlu. Metode HTTP (atau kata kerja) dan URI dari permintaan integrasi ditentukan oleh backend (yaitu, titik akhir integrasi). Mereka dapat sama dengan atau berbeda dari metode HTTP permintaan metode dan URI, masing-masing.

Misalnya, ketika fungsi Lambda mengembalikan file yang diambil dari Amazon S3, Anda dapat mengekspos operasi ini secara intuitif sebagai permintaan GET metode ke klien meskipun permintaan integrasi yang sesuai mengharuskan permintaan POST digunakan untuk menjalankan fungsi Lambda. Untuk titik akhir HTTP, kemungkinan permintaan metode dan permintaan integrasi yang sesuai keduanya menggunakan kata kerja HTTP yang sama. Namun, ini tidak diperlukan. Anda dapat mengintegrasikan permintaan metode berikut:

```
GET /{var}?query=value
Host: api.domain.net
```

Dengan permintaan integrasi berikut:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
  path: "{var}'s value",
  type: "value"
}
```

Sebagai pengembang API, Anda dapat menggunakan kata kerja HTTP dan URI apa pun untuk permintaan metode yang sesuai dengan kebutuhan Anda. Tetapi Anda harus mengikuti persyaratan titik akhir integrasi. Ketika data permintaan metode berbeda dari data permintaan integrasi, Anda dapat merekonsiliasi perbedaan dengan menyediakan pemetaan dari data permintaan metode ke data permintaan integrasi.

Dalam contoh sebelumnya, pemetaan menerjemahkan nilai variabel jalur (`{var}`) dan parameter kueri (`query`) dari permintaan GET metode ke nilai properti payload permintaan integrasi dan `path` type Data permintaan lain yang dapat dipetakan mencakup header dan badan permintaan. Ini dijelaskan dalam [the section called “Pemetaan parameter”](#).

Saat menyiapkan permintaan integrasi proxy HTTP atau HTTP, Anda menetapkan URL endpoint HTTP backend sebagai nilai URI permintaan integrasi. Misalnya, di PetStore API, permintaan metode untuk mendapatkan halaman hewan peliharaan memiliki URI permintaan integrasi berikut:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Saat menyiapkan integrasi proxy Lambda atau Lambda, Anda menetapkan Nama Sumber Daya Amazon (ARN) untuk menjalankan fungsi Lambda sebagai nilai URI permintaan integrasi. ARN ini memiliki format sebagai berikut:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

Bagian setelahnya `arn:aws:apigateway:api-region:lambda:path/`, yaitu `/2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations`, adalah jalur REST API URI dari tindakan Lambda [Invoke](#). Jika Anda menggunakan konsol API Gateway untuk menyiapkan integrasi Lambda, API Gateway akan membuat ARN dan menetakannya ke URI integrasi setelah meminta Anda untuk memilih dari suatu wilayah. *lambda-function-name*

Saat menyiapkan permintaan integrasi dengan tindakan AWS layanan lain, URI permintaan integrasi juga merupakan ARN, mirip dengan integrasi dengan tindakan Lambda. Invoke Misalnya, untuk integrasi dengan [GetBucket](#) aksi Amazon S3, URI permintaan integrasi adalah ARN dengan format berikut:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

URI permintaan integrasi adalah konvensi jalur untuk menentukan tindakan, di *{bucket}* mana placeholder nama bucket. Atau, tindakan AWS layanan dapat direferensikan dengan namanya. Menggunakan nama tindakan, URI permintaan integrasi untuk GetBucket tindakan Amazon S3 menjadi sebagai berikut:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Dengan URI permintaan integrasi berbasis tindakan, nama bucket (*{bucket}*) harus ditentukan dalam isi permintaan integrasi ({ Bucket: "*{bucket}*" }), mengikuti format input tindakanGetBucket.

Untuk AWS integrasi, Anda juga harus mengonfigurasi [kredensial](#) agar API Gateway dapat memanggil tindakan terintegrasi. Anda dapat membuat peran IAM baru atau memilih peran IAM yang sudah ada untuk API Gateway untuk memanggil tindakan dan kemudian menentukan peran menggunakan ARN-nya. Berikut ini menunjukkan contoh ARN ini:

```
arn:aws:iam::account-id:role/iam-role-name
```

Peran IAM ini harus berisi kebijakan untuk memungkinkan tindakan dieksekusi. Itu juga harus memiliki API Gateway yang dideklarasikan (dalam hubungan kepercayaan peran) sebagai entitas tepercaya untuk mengambil peran. Izin tersebut dapat diberikan pada tindakan itu sendiri. Mereka dikenal sebagai izin berbasis sumber daya. Untuk integrasi Lambda, Anda dapat memanggil tindakan [AddPermission Lambda untuk menyetel izin](#) berbasis sumber daya dan kemudian menyetel ke null dalam permintaan integrasi API Gateway. `credentials`

Kami membahas pengaturan integrasi dasar. Pengaturan lanjutan melibatkan pemetaan data permintaan metode ke data permintaan integrasi. Untuk informasi selengkapnya, lihat [the section called "Transformasi data"](#).

Pilih jenis integrasi API Gateway API

Anda memilih jenis integrasi API sesuai dengan jenis titik akhir integrasi yang bekerja dengan Anda dan bagaimana Anda ingin data diteruskan ke dan dari titik akhir integrasi. Untuk fungsi Lambda, Anda dapat memiliki integrasi proxy Lambda, atau integrasi kustom Lambda. Untuk titik akhir HTTP, Anda dapat memiliki integrasi proxy HTTP atau integrasi kustom HTTP. Untuk tindakan AWS layanan, Anda memiliki AWS integrasi tipe non-proxy saja. API Gateway juga mendukung integrasi

tiruan, di mana API Gateway berfungsi sebagai titik akhir integrasi untuk menanggapi permintaan metode.

Integrasi kustom Lambda adalah kasus khusus AWS integrasi, di mana titik akhir integrasi sesuai dengan tindakan [pemanggilan fungsi layanan Lambda](#).

Secara terprogram, Anda memilih jenis integrasi dengan mengatur [type](#) properti pada sumber daya. [Integration](#) Untuk integrasi proxy Lambda, nilainya adalah `AWS_PROXY` Untuk integrasi kustom Lambda dan semua AWS integrasi lainnya, memang demikian. `AWS` Untuk integrasi proxy HTTP dan integrasi HTTP, nilainya adalah `HTTP_PROXY` dan `HTTP`, masing-masing. Untuk integrasi tiruan, `type` nilainya adalah `MOCK`.

Integrasi proxy Lambda mendukung pengaturan integrasi yang efisien dengan satu fungsi Lambda. Pengaturannya sederhana dan dapat berkembang dengan backend tanpa harus meruntuhkan pengaturan yang ada. Untuk alasan ini, sangat disarankan untuk integrasi dengan fungsi Lambda.

Sebaliknya, integrasi kustom Lambda memungkinkan penggunaan kembali template pemetaan yang dikonfigurasi untuk berbagai titik akhir integrasi yang memiliki persyaratan serupa dari format data input dan output. Pengaturan lebih terlibat dan direkomendasikan untuk skenario aplikasi yang lebih maju.

Demikian pula, integrasi proxy HTTP memiliki pengaturan integrasi yang efisien dan dapat berkembang dengan backend tanpa harus meruntuhkan pengaturan yang ada. Integrasi kustom HTTP lebih terlibat untuk disiapkan, tetapi memungkinkan penggunaan kembali template pemetaan yang dikonfigurasi untuk titik akhir integrasi lainnya.

Daftar berikut merangkum jenis integrasi yang didukung:

- **AWS:** Jenis integrasi ini memungkinkan API mengekspos tindakan AWS layanan. Dalam AWS integrasi, Anda harus mengonfigurasi permintaan integrasi dan respons integrasi dan menyiapkan pemetaan data yang diperlukan dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode.
- **AWS_PROXY:** Jenis integrasi ini memungkinkan metode API diintegrasikan dengan tindakan pemanggilan fungsi Lambda dengan pengaturan integrasi yang fleksibel, serbaguna, dan efisien. Integrasi ini bergantung pada interaksi langsung antara klien dan fungsi Lambda terintegrasi.

Dengan jenis integrasi ini, juga dikenal sebagai integrasi proxy Lambda, Anda tidak mengatur permintaan integrasi atau respons integrasi. API Gateway meneruskan permintaan masuk dari klien sebagai input ke fungsi Lambda backend. Fungsi Lambda terintegrasi mengambil [input format](#)

[ini dan mem-parsing](#) input dari semua sumber yang tersedia, termasuk header permintaan, variabel jalur URL, parameter string kueri, dan isi yang berlaku. Fungsi mengembalikan hasil mengikuti [format output](#) ini.

Ini adalah jenis integrasi yang disukai untuk memanggil fungsi Lambda melalui API Gateway dan tidak berlaku untuk tindakan AWS layanan lainnya, termasuk tindakan Lambda selain tindakan pemanggilan fungsi.

- HTTP: Jenis integrasi ini memungkinkan API mengekspos titik akhir HTTP di backend. Dengan HTTP integrasi, juga dikenal sebagai integrasi kustom HTTP, Anda harus mengonfigurasi permintaan integrasi dan respons integrasi. Anda harus menyiapkan pemetaan data yang diperlukan dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode.
- HTTP_PROXY: Integrasi proxy HTTP memungkinkan klien untuk mengakses titik akhir HTTP backend dengan pengaturan integrasi yang efisien pada metode API tunggal. Anda tidak mengatur permintaan integrasi atau respons integrasi. API Gateway meneruskan permintaan masuk dari klien ke titik akhir HTTP dan meneruskan respons keluar dari titik akhir HTTP ke klien.
- MOCK: Jenis integrasi ini memungkinkan API Gateway mengembalikan respons tanpa mengirim permintaan lebih lanjut ke backend. Ini berguna untuk pengujian API karena dapat digunakan untuk menguji pengaturan integrasi tanpa menimbulkan biaya untuk menggunakan backend dan untuk mengaktifkan pengembangan kolaboratif API.

Dalam pengembangan kolaboratif, tim dapat mengisolasi upaya pengembangan mereka dengan menyiapkan simulasi komponen API yang dimiliki oleh tim lain dengan menggunakan integrasi. MOCK Ini juga digunakan untuk mengembalikan header terkait CORS untuk memastikan bahwa metode API mengizinkan akses CORS. Faktanya, konsol API Gateway mengintegrasikan OPTIONS metode untuk mendukung CORS dengan integrasi tiruan. [Respons gateway](#) adalah contoh lain dari integrasi tiruan.

Siapkan integrasi proxy dengan sumber daya proxy

Untuk menyiapkan integrasi proxy di API Gateway API dengan [sumber daya proxy](#), Anda melakukan tugas berikut:

- Buat sumber daya proxy dengan variabel jalur serakah dari. `{proxy+}`
- Tetapkan ANY metode pada sumber daya proxy.
- Integrasikan sumber daya dan metode dengan backend menggunakan tipe integrasi HTTP atau Lambda.

Note

Variabel jalur serakah, ANY metode, dan jenis integrasi proxy adalah fitur independen, meskipun umumnya digunakan bersama. Anda dapat mengonfigurasi metode HTTP tertentu pada sumber daya serakah atau menerapkan jenis integrasi non-proxy ke sumber daya proxy.

API Gateway memberlakukan batasan dan batasan tertentu saat menangani metode dengan integrasi proxy Lambda atau integrasi proxy HTTP. Lihat perinciannya di [the section called “Catatan penting”](#).

Note

Saat menggunakan integrasi proxy dengan passthrough, API Gateway mengembalikan `Content-Type:application/json` header default jika jenis konten payload tidak ditentukan.

Sumber daya proxy paling kuat ketika terintegrasi dengan backend menggunakan integrasi proxy HTTP atau integrasi proxy Lambda.

Integrasi proxy HTTP dengan sumber daya proxy

Integrasi proxy HTTP, yang ditunjuk oleh `HTTP_PROXY` API Gateway REST API, adalah untuk mengintegrasikan permintaan metode dengan titik akhir HTTP backend. Dengan tipe integrasi ini, API Gateway hanya meneruskan seluruh permintaan dan respons antara frontend dan backend, tunduk pada batasan dan [batasan](#) tertentu.

Note

Integrasi proxy HTTP mendukung header multi-nilai dan string kueri.

Saat menerapkan integrasi proxy HTTP ke sumber daya proxy, Anda dapat menyiapkan API untuk mengekspos sebagian atau seluruh hierarki titik akhir backend HTTP dengan satu pengaturan integrasi. Misalnya, backend situs web diatur ke dalam beberapa cabang simpul pohon dari simpul akar (`/site`) sebagai `/site/a0/a1/.../aN/site/b0/b1/.../bM`, dll. Jika Anda mengintegrasikan ANY metode pada sumber daya proxy `/api/{proxy+}` dengan titik akhir backend

dengan jalur URL/site/{proxy}, permintaan integrasi tunggal dapat mendukung operasi HTTP apa pun (GET, POST, dll.) di salah satu dari. $[a_0, a_1, \dots, a_N, b_0, b_1, \dots, b_M, \dots]$ Jika Anda menerapkan integrasi proxy ke metode HTTP tertentu, misalnya GET, permintaan integrasi yang dihasilkan bekerja dengan operasi yang ditentukan (yaitu, GET) pada salah satu node backend tersebut.

Integrasi proxy Lambda dengan sumber daya proxy

Integrasi proxy Lambda, yang ditunjuk oleh AWS_PROXY API Gateway REST API, adalah untuk mengintegrasikan permintaan metode dengan fungsi Lambda di backend. Dengan tipe integrasi ini, API Gateway menerapkan template pemetaan default untuk mengirim seluruh permintaan ke fungsi Lambda dan mengubah output dari fungsi Lambda menjadi respons HTTP.

Demikian pula, Anda dapat menerapkan integrasi proxy Lambda ke sumber daya proxy /api/{proxy+} untuk menyiapkan integrasi tunggal agar fungsi Lambda backend bereaksi satu per satu terhadap perubahan di salah satu sumber daya API di bawah. /api

Menyiapkan permintaan integrasi API menggunakan konsol API Gateway

Penyiapan metode API mendefinisikan metode dan menjelaskan perilakunya. Untuk mengatur metode, Anda harus menentukan sumber daya, termasuk root ("/"), di mana metode diekspos, metode HTTP (, dll.) GETPOST, Dan bagaimana itu akan diintegrasikan dengan backend yang ditargetkan. Permintaan dan respons metode menentukan kontrak dengan aplikasi panggilan, menetapkan parameter mana yang dapat diterima API dan seperti apa responsnya.

Prosedur berikut menjelaskan cara menggunakan konsol API Gateway untuk membuat permintaan integrasi.

Topik

- [Siapkan integrasi Lambda](#)
- [Siapkan integrasi HTTP](#)
- [Siapkan integrasi AWS layanan](#)
- [Siapkan integrasi tiruan](#)

Siapkan integrasi Lambda

Gunakan integrasi fungsi Lambda untuk mengintegrasikan API Anda dengan fungsi Lambda. Pada API level, ini adalah tipe AWS integrasi jika Anda membuat integrasi non-proxy, atau tipe AWS_PROXY integrasi jika Anda membuat integrasi proxy.

Untuk mengatur integrasi Lambda

1. Di panel Resources, pilih Create method.
2. Untuk jenis Metode, pilih metode HTTP.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Untuk menggunakan integrasi proxy Lambda, aktifkan integrasi proxy Lambda. Untuk mempelajari lebih lanjut tentang integrasi proxy Lambda, lihat. [the section called “Memahami integrasi proxy Lambda”](#)
5. Untuk fungsi Lambda, masukkan nama fungsi Lambda.

Jika Anda menggunakan fungsi Lambda di Wilayah yang berbeda dari API Anda, pilih Wilayah dari menu tarik-turun dan masukkan nama fungsi Lambda. Jika Anda menggunakan fungsi Lambda lintas akun, masukkan fungsi ARN.

6. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000
7. (Opsional) Anda dapat mengonfigurasi pengaturan permintaan metode menggunakan menu tarik-turun berikut. Pilih Pengaturan permintaan metode dan konfigurasi permintaan metode Anda. Untuk informasi lebih lanjut, lihat langkah 3 dari [the section called “Mengedit permintaan metode di konsol”](#).

Anda juga dapat mengonfigurasi pengaturan permintaan metode Anda setelah Anda membuat metode Anda.

8. Pilih metode Buat.

Siapkan integrasi HTTP

Gunakan integrasi HTTP untuk mengintegrasikan API Anda dengan titik akhir HTTP. Pada level API, ini adalah tipe HTTP integrasi.

Untuk mengatur integrasi HTTP

1. Di panel Resources, pilih Create method.
2. Untuk jenis Metode, pilih metode HTTP.
3. Untuk jenis Integrasi, pilih HTTP.

4. Untuk menggunakan integrasi proxy HTTP, aktifkan integrasi proxy HTTP. Untuk mempelajari lebih lanjut tentang integrasi proxy HTTP, lihat [the section called “Siapkan integrasi proxy HTTP di API Gateway”](#).
5. Untuk metode HTTP, pilih jenis metode HTTP yang paling cocok dengan metode di backend HTTP.
6. Untuk URL Endpoint, masukkan URL backend HTTP yang ingin digunakan metode ini.
7. Untuk penanganan Konten, pilih perilaku penanganan konten.
8. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000
9. (Opsional) Anda dapat mengonfigurasi pengaturan permintaan metode menggunakan menu tarik-turun berikut. Pilih Pengaturan permintaan metode dan konfigurasi permintaan metode Anda. Untuk informasi lebih lanjut, lihat langkah 3 dari [the section called “Mengedit permintaan metode di konsol”](#).

Anda juga dapat mengonfigurasi pengaturan permintaan metode Anda setelah Anda membuat metode Anda.

10. Pilih metode Buat.

Siapkan integrasi AWS layanan

Gunakan integrasi AWS layanan untuk mengintegrasikan API Anda secara langsung dengan AWS layanan. Pada level API, ini adalah tipe AWS integrasi.

Untuk menyiapkan API Gateway API untuk melakukan salah satu hal berikut:

- Buat fungsi Lambda baru.
- Tetapkan izin sumber daya pada fungsi Lambda.
- Lakukan tindakan layanan Lambda lainnya.

Anda harus memilih AWS layanan.

Untuk mengatur integrasi AWS layanan

1. Di panel Resources, pilih Create method.
2. Untuk jenis Metode, pilih metode HTTP.

3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk AWS Wilayah, pilih AWS Wilayah yang ingin digunakan metode ini untuk memanggil tindakan.
5. Untuk AWS layanan, pilih AWS layanan yang Anda inginkan untuk memanggil metode ini.
6. Untuk AWS subdomain, masukkan subdomain yang digunakan oleh layanan. AWS Biasanya, Anda akan membiarkan ini kosong. Beberapa AWS layanan dapat mendukung subdomain sebagai bagian dari host. Konsultasikan dokumentasi layanan untuk ketersediaan dan, jika tersedia, detailnya.
7. Untuk metode HTTP, pilih jenis metode HTTP yang sesuai dengan tindakan. Untuk jenis metode HTTP, lihat dokumentasi referensi API untuk AWS layanan yang Anda pilih untuk AWS layanan.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan untuk menggunakan tindakan API atau Use path override untuk menggunakan jalur sumber daya kustom. Untuk tindakan yang tersedia dan jalur sumber daya khusus, lihat dokumentasi referensi API untuk AWS layanan yang Anda pilih untuk AWS layanan.
9. Masukkan nama Action atau Path override.
10. Untuk peran Eksekusi, masukkan ARN dari peran IAM yang akan digunakan metode untuk memanggil tindakan.

Untuk membuat peran IAM, Anda dapat menyesuaikan instruksi di [the section called “Langkah 1: Buat peran eksekusi proxy AWS layanan”](#). Tentukan kebijakan akses dengan jumlah tindakan dan pernyataan sumber daya yang diinginkan. Untuk informasi selengkapnya, lihat [the section called “Cara kerja Amazon API Gateway dengan IAM”](#).

Untuk sintaks pernyataan tindakan dan sumber daya, lihat dokumentasi untuk AWS layanan yang Anda pilih untuk AWS layanan.

Untuk hubungan kepercayaan peran IAM, tentukan hal berikut, yang memungkinkan API Gateway mengambil tindakan atas nama AWS akun Anda:

JSON

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

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

11. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000
12. (Opsional) Anda dapat mengonfigurasi pengaturan permintaan metode menggunakan menu tarik-turun berikut. Pilih Pengaturan permintaan metode dan konfigurasi permintaan metode Anda. Untuk informasi lebih lanjut, lihat langkah 3 dari [the section called “Mengedit permintaan metode di konsol”](#).

Anda juga dapat mengonfigurasi pengaturan permintaan metode Anda setelah Anda membuat metode Anda.

13. Pilih metode Buat.

Siapkan integrasi tiruan

Gunakan integrasi tiruan jika Anda ingin API Gateway bertindak sebagai backend Anda untuk mengembalikan respons statis. Pada level API, ini adalah tipe MOCK integrasi. Biasanya, Anda dapat menggunakan MOCK integrasi saat API Anda belum final, tetapi Anda ingin menghasilkan respons API untuk membuka blokir tim dependen untuk pengujian. Untuk OPTION metode ini, API Gateway menetapkan MOCK integrasi sebagai default untuk mengembalikan header yang mengaktifkan CORS untuk sumber daya API yang diterapkan.

Untuk mengatur integrasi tiruan

1. Di panel Resources, pilih Create method.
2. Untuk jenis Metode, pilih metode HTTP.
3. Untuk tipe Integrasi, pilih Mock.
4. (Opsional) Anda dapat mengonfigurasi pengaturan permintaan metode menggunakan menu tarik-turun berikut. Pilih Pengaturan permintaan metode dan konfigurasi permintaan metode Anda. Untuk informasi lebih lanjut, lihat langkah 3 dari [the section called “Mengedit permintaan metode di konsol”](#).

Anda juga dapat mengonfigurasi pengaturan permintaan metode Anda setelah Anda membuat metode Anda.

5. Pilih metode Buat.

Siapkan respons integrasi di API Gateway

Untuk integrasi non-proxy, Anda harus menyiapkan setidaknya satu respons integrasi, dan menjadikannya respons default, untuk meneruskan hasil yang dikembalikan dari backend ke klien. Anda dapat memilih untuk melewati hasil apa adanya atau mengubah data respons integrasi ke data respons metode jika keduanya memiliki format yang berbeda.

Untuk integrasi proxy, API Gateway secara otomatis meneruskan output backend ke klien sebagai respons HTTP. Anda tidak menetapkan respons integrasi atau respons metode.

Untuk menyiapkan respons integrasi, Anda melakukan tugas wajib dan opsional berikut:

1. Tentukan kode status HTTP dari respons metode yang data respons integrasi dipetakan. Ini wajib diisi.
2. Tentukan ekspresi reguler untuk memilih output backend yang akan diwakili oleh respons integrasi ini. Jika Anda membiarkan ini kosong, responsnya adalah respons default yang digunakan untuk menangkap respons apa pun yang belum dikonfigurasi.
3. Jika diperlukan, deklarasikan pemetaan yang terdiri dari pasangan nilai kunci untuk memetakan parameter respons integrasi yang ditentukan ke parameter respons metode yang diberikan.
4. Jika perlu, tambahkan templat pemetaan tubuh untuk mengubah muatan respons integrasi yang diberikan menjadi muatan respons metode yang ditentukan.
5. Jika diperlukan, tentukan cara menangani konversi tipe untuk muatan biner.

Respons integrasi adalah respons HTTP yang merangkum respons backend. Untuk titik akhir HTTP, respons backend adalah respons HTTP. Kode status respons integrasi dapat mengambil kode status yang dikembalikan ke backend, dan badan respons integrasi adalah payload yang dikembalikan ke backend. Untuk titik akhir Lambda, respons backend adalah output yang dikembalikan dari fungsi Lambda. Dengan integrasi Lambda, output fungsi Lambda dikembalikan sebagai respons. 200 OK Payload dapat berisi hasil sebagai data JSON, termasuk string JSON atau objek JSON, atau pesan kesalahan sebagai objek JSON. Anda dapat menetapkan ekspresi reguler ke properti [SelectionPattern](#) untuk memetakan respons kesalahan terhadap respons kesalahan HTTP yang sesuai. Untuk informasi selengkapnya tentang respons kesalahan fungsi Lambda, lihat. [Menangani](#)

[kesalahan Lambda di API Gateway](#) Dengan integrasi proxy Lambda, fungsi Lambda harus mengembalikan output dari format berikut:

```
{
  statusCode: "...",           // a valid HTTP status code
  headers: {
    custom-header: "..."     // any API-specific custom header
  },
  body: "...",                 // a JSON string.
  isBase64Encoded: true|false // for binary support
}
```

Tidak perlu memetakan respons fungsi Lambda ke respons HTTP yang tepat.

Untuk mengembalikan hasil ke klien, atur respons integrasi untuk meneruskan respons titik akhir melalui apa adanya ke respons metode yang sesuai. Atau Anda dapat memetakan data respons titik akhir ke data respons metode. Data respons yang dapat dipetakan mencakup kode status respons, parameter header respons, dan badan respons. Jika tidak ada respons metode yang ditentukan untuk kode status yang dikembalikan, API Gateway mengembalikan kesalahan 500. Lihat informasi yang lebih lengkap di [Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway](#).

Integrasi Lambda untuk REST APIs di API Gateway

Anda dapat mengintegrasikan metode API dengan fungsi Lambda menggunakan integrasi proxy Lambda atau integrasi Lambda non-proxy (kustom).

Dalam integrasi proxy Lambda, pengaturan yang diperlukan sederhana. Setel metode HTTP integrasi ke POST, URI endpoint integrasi ke ARN dari tindakan pemanggilan fungsi Lambda dari fungsi Lambda tertentu, dan berikan izin API Gateway untuk memanggil fungsi Lambda atas nama Anda.

Di integrasi non-proxy Lambda, selain langkah penyiapan integrasi proxy, Anda juga menentukan cara data permintaan masuk dipetakan ke permintaan integrasi dan bagaimana data respons integrasi yang dihasilkan dipetakan ke respons metode.

Topik

- [Integrasi proxy Lambda di API Gateway](#)
- [Siapkan integrasi kustom Lambda di API Gateway](#)
- [Siapkan pemanggilan asinkron dari fungsi Lambda backend](#)

- [Menangani kesalahan Lambda di API Gateway](#)

Integrasi proxy Lambda di API Gateway

Bagian berikut menunjukkan cara menggunakan integrasi proxy Lambda.

Topik

- [Memahami integrasi proxy API Gateway Lambda](#)
- [Support untuk header multi-nilai dan parameter string kueri](#)
- [Format input fungsi Lambda untuk integrasi proxy](#)
- [Format output dari fungsi Lambda untuk integrasi proxy](#)
- [Siapkan integrasi proxy Lambda untuk API Gateway menggunakan AWS CLI](#)
- [Siapkan sumber daya proxy dengan integrasi proxy Lambda dengan definisi OpenAPI](#)

Memahami integrasi proxy API Gateway Lambda

Integrasi proxy Amazon API Gateway Lambda adalah mekanisme yang sederhana, kuat, dan gesit untuk membangun API dengan penyiapan metode API tunggal. Integrasi proxy Lambda memungkinkan klien untuk memanggil satu fungsi Lambda di backend. Fungsi ini mengakses banyak sumber daya atau fitur AWS layanan lain, termasuk memanggil fungsi Lambda lainnya.

Dalam integrasi proxy Lambda, ketika klien mengirimkan permintaan API, API Gateway meneruskan ke fungsi Lambda terintegrasi [objek peristiwa](#), kecuali urutan parameter permintaan tidak dipertahankan. [Data permintaan](#) ini mencakup header permintaan, parameter string kueri, variabel jalur URL, payload, dan data konfigurasi API. Data konfigurasi dapat mencakup nama tahap penyebaran saat ini, variabel tahap, identitas pengguna, atau konteks otorisasi (jika ada). Fungsi backend Lambda mem-parsing data permintaan yang masuk untuk menentukan respons yang dikembalikan. [Agar API Gateway meneruskan output Lambda sebagai respons API ke klien, fungsi Lambda harus mengembalikan hasilnya dalam format ini.](#)

Karena API Gateway tidak terlalu banyak campur tangan antara klien dan fungsi Lambda backend untuk integrasi proxy Lambda, klien dan fungsi Lambda terintegrasi dapat beradaptasi dengan perubahan satu sama lain tanpa merusak pengaturan integrasi API yang ada. Untuk mengaktifkan ini, klien harus mengikuti protokol aplikasi yang diberlakukan oleh fungsi Lambda backend.

Anda dapat menyiapkan integrasi proxy Lambda untuk metode API apa pun. Tetapi integrasi proxy Lambda lebih kuat ketika dikonfigurasi untuk metode API yang melibatkan sumber daya proxy

generik. Sumber daya proxy generik dapat dilambangkan dengan variabel jalur template khusus, placeholder metode catch-all `{proxy+}` ANY, atau keduanya. Klien dapat meneruskan input ke fungsi Lambda backend dalam permintaan masuk sebagai parameter permintaan atau muatan yang berlaku. Parameter permintaan termasuk header, variabel jalur URL, parameter string kueri, dan payload yang berlaku. Fungsi Lambda terintegrasi memverifikasi semua sumber input sebelum memproses permintaan dan menanggapi klien dengan pesan kesalahan yang berarti jika ada input yang diperlukan yang hilang.

Saat memanggil metode API yang terintegrasi dengan metode HTTP generik ANY dan sumber daya generik `{proxy+}`, klien mengirimkan permintaan dengan metode HTTP tertentu sebagai pengganti. ANY Klien juga menentukan jalur URL tertentu, bukan `{proxy+}`, dan menyertakan header yang diperlukan, parameter string kueri, atau payload yang berlaku.

Daftar berikut merangkum perilaku runtime dari berbagai metode API dengan integrasi proxy Lambda:

- ANY `/ {proxy+}`: Klien harus memilih metode HTTP tertentu, harus menetapkan hierarki jalur sumber daya tertentu, dan dapat mengatur header, parameter string kueri, dan muatan yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- ANY `/res`: Klien harus memilih metode HTTP tertentu dan dapat mengatur header, parameter string kueri, dan payload yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- GET|POST|PUT|... `/ {proxy+}`: Klien dapat mengatur hierarki jalur sumber daya tertentu, header apa pun, parameter string kueri, dan muatan yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- GET|POST|PUT|... `/res/{path}/...`: Klien harus memilih segmen jalur tertentu (untuk `{path}` variabel) dan dapat mengatur header permintaan, parameter string kueri, dan payload yang berlaku untuk meneruskan data input ke fungsi Lambda terintegrasi.
- GET|POST|PUT|... `/res`: Klien dapat memilih header permintaan, parameter string kueri, dan payload yang berlaku untuk meneruskan data input ke fungsi Lambda terintegrasi.

Baik sumber daya proxy `{proxy+}` dan sumber daya kustom `{custom}` dinyatakan sebagai variabel jalur templat. Namun `{proxy+}` dapat merujuk ke sumber daya apa pun di sepanjang hierarki jalur, sementara `{custom}` mengacu pada segmen jalur tertentu saja. Misalnya, toko kelontong mungkin mengatur inventaris produk online-nya berdasarkan nama departemen, kategori produksi, dan jenis produk. Situs web toko kelontong kemudian dapat mewakili produk yang tersedia

dengan variabel jalur template berikut dari sumber daya khusus: `/ {department} / {produce-category} / {product-type}` Misalnya, apel diwakili oleh `/produce/fruit/apple` dan wortel oleh `/produce/vegetables/carrot`. Ini juga dapat digunakan `/ {proxy+}` untuk mewakili departemen apa pun, kategori produk apa pun, atau jenis produk apa pun yang dapat dicari pelanggan saat berbelanja di toko online. Misalnya, `/ {proxy+}` dapat merujuk ke salah satu item berikut:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Untuk memungkinkan pelanggan mencari produk apa pun yang tersedia, kategori produksinya, dan departemen toko terkait, Anda dapat mengekspos satu metode GET `/ {proxy+}` dengan izin hanya-baca. Demikian pula, untuk memungkinkan supervisor memperbarui inventaris produce departemen, Anda dapat menyiapkan metode tunggal lain PUT `/produce/ {proxy+}` dengan read/write izin. Untuk memungkinkan kasir memperbarui total sayuran yang sedang berjalan, Anda dapat mengatur POST `/produce/vegetables/ {proxy+}` metode dengan read/write izin. Untuk membiarkan manajer toko melakukan tindakan apa pun yang mungkin pada produk apa pun yang tersedia, pengembang toko online dapat mengekspos ANY `/ {proxy+}` metode dengan read/write izin. Dalam kasus apa pun, pada waktu berjalan, pelanggan atau karyawan harus memilih produk tertentu dari jenis tertentu di departemen yang dipilih, kategori produk tertentu di departemen yang dipilih, atau departemen tertentu.

Untuk informasi selengkapnya tentang menyiapkan integrasi proxy API Gateway, lihat [Siapkan integrasi proxy dengan sumber daya proxy](#).

Integrasi proxy mengharuskan klien memiliki pengetahuan yang lebih rinci tentang persyaratan backend. Oleh karena itu, untuk memastikan kinerja aplikasi dan pengalaman pengguna yang optimal, pengembang backend harus berkomunikasi dengan jelas kepada pengembang klien persyaratan backend, dan memberikan mekanisme umpan balik kesalahan yang kuat ketika persyaratan tidak terpenuhi.

Support untuk header multi-nilai dan parameter string kueri

API Gateway mendukung beberapa header dan parameter string kueri yang memiliki nama yang sama. Header multi-nilai serta header dan parameter nilai tunggal dapat digabungkan dalam

permintaan dan tanggapan yang sama. Untuk informasi selengkapnya, lihat [Format input fungsi Lambda untuk integrasi proxy](#) dan [Format output dari fungsi Lambda untuk integrasi proxy](#).

Format input fungsi Lambda untuk integrasi proxy

Dalam integrasi proxy Lambda, API Gateway memetakan seluruh permintaan klien ke event parameter input fungsi Lambda backend. Contoh berikut menunjukkan struktur peristiwa yang dikirimkan API Gateway ke integrasi proxy Lambda.

Dalam contoh ini, kami berasumsi bahwa pemanggilan ke API Gateway adalah sebagai berikut:

```
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'
```

Outputnya terlihat seperti berikut:

```
{
  "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",
    "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

Dalam masukan:

- `headersKuncinya` hanya dapat berisi header nilai tunggal.
- `multiValueHeadersKuncinya` dapat berisi header multi-nilai serta header nilai tunggal.
- Jika Anda menentukan nilai untuk keduanya `headers` dan `multiValueHeaders`, API Gateway menggabungkannya ke dalam satu daftar. Jika pasangan kunci-nilai yang sama ditentukan di keduanya, hanya nilai dari yang `multiValueHeaders` akan muncul dalam daftar gabungan.

Dalam input ke fungsi Lambda backend, `requestContext` objek adalah peta pasangan kunci-nilai. Di setiap pasangan, kuncinya adalah nama properti variabel [\\$context](#), dan nilainya adalah nilai properti itu. API Gateway mungkin menambahkan kunci baru ke peta.

Bergantung pada fitur yang diaktifkan, `requestContext` peta dapat bervariasi dari API ke API. Misalnya, dalam contoh sebelumnya, tidak ada jenis otorisasi yang ditentukan, jadi tidak ada `$context.authorizer.*` atau `$context.identity.*` properti yang ada. Ketika jenis otorisasi ditentukan, ini menyebabkan API Gateway meneruskan informasi pengguna yang diotorisasi ke titik akhir integrasi dalam `requestContext.identity` objek sebagai berikut:

- Ketika jenis otorisasi `AWS_IAM`, informasi pengguna yang berwenang mencakup `$context.identity.*` properti.
- Ketika jenis otorisasi adalah `COGNITO_USER_POOLS` (Amazon Cognito Authorizer), informasi `$context.identity.cognito*` pengguna yang berwenang termasuk dan properti `$context.authorizer.claims.*`
- Ketika jenis otorisasi adalah `CUSTOM` (Lambda Authorizer), informasi pengguna yang berwenang `$context.authorizer.principalId` termasuk dan properti lain yang berlaku `$context.authorizer.*`

Format output dari fungsi Lambda untuk integrasi proxy

Dalam integrasi proxy Lambda, API Gateway memerlukan fungsi Lambda backend untuk mengembalikan output sesuai dengan format JSON berikut:

```
{
  "isBase64Encoded": true/false,
  "statusCode": httpStatusCode,
  "headers": { "headerName": "headerValue", ... },
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "body": "... "
}
```

Dalam output:

- `multiValueHeaders` Tombol headers dan dapat tidak ditentukan jika tidak ada header respons tambahan yang akan dikembalikan.
- `headers` Kuncinya hanya dapat berisi header nilai tunggal.
- `multiValueHeaders` Kuncinya dapat berisi header multi-nilai serta header nilai tunggal. Anda dapat menggunakan `multiValueHeaders` kunci untuk menentukan semua header tambahan Anda, termasuk yang bernilai tunggal.
- Jika Anda menentukan nilai untuk keduanya `headers` dan `multiValueHeaders`, API Gateway menggabungkannya ke dalam satu daftar. Jika pasangan kunci-nilai yang sama ditentukan di keduanya, hanya nilai dari yang `multiValueHeaders` akan muncul dalam daftar gabungan.

Untuk mengaktifkan CORS untuk integrasi proxy Lambda, Anda harus `Access-Control-Allow-Origin: domain-name` menambahkan ke output. `headers` *domain-name* bisa * untuk nama domain apa pun. Output body disusun ke frontend sebagai payload respons metode. Jika **body** adalah gumpalan biner, Anda dapat menyandikannya sebagai string yang dikodekan Base64 dengan mengatur **true** dan **isBase64Encoded** mengonfigurasi `*/*` sebagai Tipe Media Biner. Jika tidak, Anda dapat mengaturnya ke `false` atau membiarkannya tidak ditentukan.

Note

Untuk informasi selengkapnya tentang mengaktifkan dukungan biner, lihat [Mengaktifkan dukungan biner menggunakan konsol API Gateway](#). Untuk contoh fungsi Lambda, lihat [Kembalikan media biner dari integrasi proxy Lambda di API Gateway](#)

Jika output fungsi dari format yang berbeda, API Gateway mengembalikan respon 502 Bad Gateway kesalahan.

Untuk mengembalikan respons dalam fungsi Lambda di Node.js, Anda dapat menggunakan perintah seperti berikut:

- Untuk mengembalikan hasil yang sukses, `callback(null, {"statusCode": 200, "body": "results"})`.
- Untuk melempar pengecualian, `callback(new Error('internal server error'))`.
- Untuk kesalahan sisi klien (jika, misalnya, parameter yang diperlukan tidak ada), Anda dapat memanggil `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` untuk mengembalikan kesalahan tanpa melempar pengecualian.

Dalam async fungsi Lambda di Node.js, sintaks yang setara adalah:

- Untuk mengembalikan hasil yang sukses, `return {"statusCode": 200, "body": "results"}`.
- Untuk melempar pengecualian, `throw new Error("internal server error")`.
- Untuk kesalahan sisi klien (jika, misalnya, parameter yang diperlukan tidak ada), Anda dapat memanggil `return {"statusCode": 400, "body": "Missing parameters of ..."}` untuk mengembalikan kesalahan tanpa melempar pengecualian.

Siapkan integrasi proxy Lambda untuk API Gateway menggunakan AWS CLI

Di bagian ini, kami menunjukkan cara menyiapkan API dengan integrasi proxy Lambda menggunakan AWS CLI Untuk petunjuk mendetail tentang penggunaan konsol API Gateway guna mengonfigurasi sumber daya proxy dengan integrasi proxy Lambda, lihat. [Tutorial: Buat REST API dengan integrasi proxy Lambda](#)

Sebagai contoh, kami menggunakan contoh fungsi Lambda berikut sebagai backend 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
};

```

Membandingkan ini dengan pengaturan integrasi kustom Lambda di [the section called “Siapkan integrasi kustom Lambda”](#), input ke fungsi Lambda ini dapat dinyatakan dalam parameter permintaan dan isi. Anda memiliki lebih banyak garis lintang untuk memungkinkan klien meneruskan data input yang sama. Di sini, klien dapat meneruskan nama penyambut sebagai parameter string kueri, header, atau properti tubuh. Fungsi ini juga dapat mendukung integrasi kustom Lambda. Penyiapan API lebih sederhana. Anda tidak mengonfigurasi respons metode atau respons integrasi sama sekali.

Untuk mengatur integrasi proxy Lambda menggunakan AWS CLI

1. Gunakan [create-rest-api](#) perintah berikut untuk membuat API:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

Outputnya akan terlihat seperti berikut:

```
{
  "name": "HelloWorldProxy (AWS CLI)",
  "id": "te6si5ach7",

```



```
"rootResourceId" : "krznpq9xpg",
"createdDate": 1508461860
}
```

Anda menggunakan API id (te6si5ach7) dan rootResourceId (krznpq9xpg) di seluruh contoh ini.

2. [Gunakan perintah create-resource berikut untuk membuat API Gateway Resource dari: /greeting](#)

```
aws apigateway create-resource \
  --rest-api-id te6si5ach7 \
  --parent-id krznpq9xpg \
  --path-part {proxy+}
```

Outputnya akan terlihat seperti berikut:

```
{
  "path": "/{proxy+}",
  "pathPart": "{proxy+}",
  "id": "2jf6xt",
  "parentId": "krznpq9xpg"
}
```

Anda menggunakan id nilai {proxy+} sumber daya (2jf6xt) untuk membuat metode pada /{proxy+} sumber daya di langkah berikutnya.

3. Gunakan [put-metode](#) berikut untuk membuat permintaan ANY metode: ANY /{proxy+}

```
aws apigateway put-method --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method ANY \
  --authorization-type "NONE"
```

Outputnya akan terlihat seperti berikut:

```
{
  "apiKeyRequired": false,
  "httpMethod": "ANY",
  "authorizationType": "NONE"
}
```

Metode API ini memungkinkan klien untuk menerima atau mengirim salam dari fungsi Lambda di backend.

- Gunakan perintah [put-integration](#) berikut untuk mengatur integrasi ANY /{proxy+} metode dengan fungsi Lambda, bernama. HelloWorld Fungsi ini menanggapi permintaan dengan pesan "Hello, {name}!", jika greeter parameter disediakan, atau "Hello, World!", jika parameter string query tidak diatur.

```
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

Untuk integrasi Lambda, Anda harus menggunakan metode HTTP POST untuk permintaan integrasi, sesuai dengan [spesifikasi tindakan layanan Lambda untuk pemanggilan](#) fungsi. Peran IAM apigAwsProxyRole harus memiliki kebijakan yang memungkinkan apigateway layanan untuk menjalankan fungsi Lambda. Untuk informasi selengkapnya tentang izin IAM, lihat. [the section called “Model izin API Gateway untuk menjalankan API”](#)

Outputnya akan terlihat seperti berikut:

```
{  
  "passthroughBehavior": "WHEN_NO_MATCH",  
  "cacheKeyParameters": [],  
  "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/  
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations",  
  "httpMethod": "POST",  
  "cacheNamespace": "vvom7n",  
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",  
  "type": "AWS_PROXY"
```

```
}
```

Alih-alih menyediakan peran `IAMcredentials`, Anda dapat menggunakan perintah [add-permission untuk menambahkan izin berbasis](#) sumber daya. Inilah yang dilakukan konsol API Gateway.

- Gunakan perintah [create-deployment](#) berikut untuk menyebarkan API ke sebuah panggung: `test`

```
aws apigateway create-deployment \  
  --rest-api-id te6si5ach7 \  
  --stage-name test
```

- Uji API menggunakan perintah `cURL` berikut di terminal.

Memanggil API dengan parameter string kueri `?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?  
greeter=jane'
```

Memanggil API dengan parameter header `greeter:jane`:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \  
  -H 'content-type: application/json' \  
  -H 'greeter: jane'
```

Memanggil API dengan badan `{"greeter": "jane"}`:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \  
  -H 'content-type: application/json' \  
  -d '{ "greeter": "jane" }'
```

Dalam semua kasus, outputnya adalah respons 200 dengan badan respons berikut:

```
Hello, jane!
```

Siapkan sumber daya proxy dengan integrasi proxy Lambda dengan definisi OpenAPI

Untuk menyiapkan sumber daya proxy dengan tipe integrasi proxy Lambda, buat sumber daya API dengan parameter jalur serakah (misalnya, /parent/{proxy+}) dan integrasikan sumber daya ini dengan backend fungsi Lambda (misalnya,) pada metode. `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource` ANY Parameter jalur serakah harus berada di akhir jalur sumber daya API. Seperti sumber daya non-proxy, Anda dapat mengatur sumber daya proxy dengan menggunakan konsol API Gateway, mengimpor file definisi OpenAPI, atau memanggil API REST API Gateway API secara langsung.

File definisi API OpenAPI berikut menunjukkan contoh API dengan sumber daya proxy yang terintegrasi dengan fungsi Lambda bernama. `SimpleLambda4ProxyResource`

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://gy415nuihc.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": "gy415nuihc.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"
    }
}
}
}
}
}

```

Dalam integrasi proxy Lambda, pada waktu berjalan, API Gateway memetakan permintaan masuk ke event parameter input fungsi Lambda. Input mencakup metode permintaan, jalur, header, parameter string kueri apa pun, payload apa pun, konteks terkait, dan variabel tahap yang ditentukan. Format input dijelaskan dalam [Format input fungsi Lambda untuk integrasi proxy](#). Agar API Gateway berhasil memetakan output Lambda ke respons HTTP, fungsi Lambda harus menampilkan hasil dalam format yang dijelaskan dalam [Format output dari fungsi Lambda untuk integrasi proxy](#).

Dalam integrasi proxy Lambda dari sumber daya proxy melalui ANY metode, fungsi Lambda backend tunggal berfungsi sebagai event handler untuk semua permintaan melalui sumber daya proxy. Misalnya, untuk mencatat pola lalu lintas, Anda dapat meminta perangkat seluler mengirim informasi lokasi negara bagian, kota, jalan, dan bangunannya dengan mengirimkan permintaan `/state/city/street/house` di jalur URL untuk sumber daya proxy. Fungsi backend Lambda kemudian dapat mengurai jalur URL dan menyisipkan tupel lokasi ke dalam tabel DynamoDB.

Siapkan integrasi kustom Lambda di API Gateway

Untuk menunjukkan cara mengatur integrasi kustom Lambda, atau non-proxy, kami membuat API Gateway API untuk mengekspos `GET /greeting?greeter={name}` metode untuk menjalankan fungsi Lambda. Gunakan salah satu contoh fungsi Lambda berikut untuk API Anda.

Gunakan salah satu contoh fungsi Lambda berikut:

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
```

Fungsi merespons dengan pesan "Hello, {name}!" jika nilai greeter parameter adalah string yang tidak kosong. Ia mengembalikan pesan "Hello, World!" jika greeter nilai adalah string kosong. Fungsi mengembalikan pesan kesalahan "Missing the required greeter parameter." jika parameter penyambut tidak diatur dalam permintaan masuk. Kami menamai fungsinya HelloWorld.

Anda dapat membuatnya di konsol Lambda atau dengan menggunakan AWS CLI Pada bagian ini, kami mereferensikan fungsi ini menggunakan ARN berikut:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Dengan fungsi Lambda diatur di backend, lanjutkan untuk mengatur API.

Untuk mengatur integrasi kustom Lambda menggunakan AWS CLI

1. Gunakan [create-rest-api](#) perintah berikut untuk membuat API:


```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

Outputnya akan terlihat seperti berikut:

```
{
  "name": "HelloWorld (AWS CLI)",
  "id": "te6si5ach7",
  "rootResourceId" : "krznpq9xpg",
  "createdDate": 1508461860
}
```

Anda menggunakan API id (te6si5ach7) dan rootResourceId (krznpq9xpg) di seluruh contoh ini.

2. [Gunakan perintah create-resource berikut untuk membuat API Gateway Resource dari: /greeting](#)

```
aws apigateway create-resource \
  --rest-api-id te6si5ach7 \
  --parent-id krznpq9xpg \
  --path-part greeting
```

Outputnya akan terlihat seperti berikut:

```
{
  "path": "/greeting",
  "pathPart": "greeting",
  "id": "2jf6xt",
  "parentId": "krznpq9xpg"
}
```

Anda menggunakan id nilai greeting sumber daya (2jf6xt) untuk membuat metode pada /greeting sumber daya di langkah berikutnya.

3. Gunakan perintah [put-method](#) berikut untuk membuat permintaan metode API dari: GET /greeting?greeter={name}

```
aws apigateway put-method --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method GET \
```

```
--authorization-type "NONE" \
--request-parameters method.request.querystring.greeter=false
```

Outputnya akan terlihat seperti berikut:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "requestParameters": {
    "method.request.querystring.greeter": false
  }
}
```

Metode API ini memungkinkan klien untuk menerima salam dari fungsi Lambda di backend. greeterParameternya opsional karena backend harus menangani penelepon anonim atau penelepon yang diidentifikasi sendiri.

- Gunakan [put-method-response](#) perintah berikut untuk mengatur 200 OK respons terhadap permintaan metode GET /greeting?greeter={name}:

```
aws apigateway put-method-response \
  --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method GET \
  --status-code 200
```

- Gunakan perintah [put-integration](#) berikut untuk mengatur integrasi GET /greeting?greeter={name} metode dengan fungsi Lambda, bernama. HelloWorld Fungsi menanggapi permintaan dengan pesan "Hello, {name}!", jika greeter parameter disediakan, atau "Hello, World!", jika parameter string query tidak diatur.

```
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":{"$input.params('greeter')\'}}' \
--credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Template pemetaan yang disediakan di sini menerjemahkan parameter string greeter ke greeter properti payload JSON. Ini diperlukan karena input ke fungsi Lambda harus diekspresikan dalam tubuh.

Important

Untuk integrasi Lambda, Anda harus menggunakan metode HTTP POST untuk permintaan integrasi, sesuai dengan [spesifikasi tindakan layanan Lambda untuk pemanggilan](#) fungsi. uriParameternya adalah ARN dari tindakan pemanggilan fungsi. Outputnya akan terlihat seperti berikut:

```
{
  "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": "{$input.params('greeter')\'}"
  },
  "cacheNamespace": "krznpq9xpg",
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "type": "AWS"
}
```

Peran IAM apigAwsProxyRole harus memiliki kebijakan yang memungkinkan apigateway layanan untuk menjalankan fungsi Lambda. Alih-alih menyediakan peran IAMcredentials, Anda dapat memanggil perintah [add-permission untuk menambahkan izin berbasis sumber daya](#). Beginilah cara konsol API Gateway menambahkan izin ini.

- Gunakan [put-integration-response](#) perintah berikut untuk mengatur respons integrasi untuk meneruskan output fungsi Lambda ke klien sebagai respons 200 OK metode:

```
aws apigateway put-integration-response \
--rest-api-id te6si5ach7 \
```

```
--resource-id 2jf6xt \  
--http-method GET \  
--status-code 200 \  
--selection-pattern ""
```

Dengan mengatur pola pilihan ke string kosong, 200 OK responsnya adalah default.

Outputnya akan terlihat seperti berikut:

```
{  
  "selectionPattern": "",  
  "statusCode": "200"  
}
```

7. Gunakan perintah [create-deployment](#) berikut untuk menyebarkan API ke sebuah panggung: test

```
aws apigateway create-deployment \  
  --rest-api-id te6si5ach7 \  
  --stage-name test
```

8. Uji API menggunakan perintah cURL berikut di 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'
```

Siapkan pemanggilan asinkron dari fungsi Lambda backend

Dalam integrasi Lambda non-proxy (kustom), fungsi Lambda backend dipanggil secara sinkron secara default. Ini adalah perilaku yang diinginkan untuk sebagian besar operasi REST API. Beberapa aplikasi, bagaimanapun, memerlukan pekerjaan yang harus dilakukan secara asinkron (sebagai operasi batch atau operasi latensi panjang), biasanya oleh komponen backend terpisah. Dalam hal ini, fungsi Lambda backend dipanggil secara asinkron, dan metode REST API front-end tidak mengembalikan hasilnya.

Anda dapat mengonfigurasi fungsi Lambda agar integrasi non-proxy Lambda dipanggil secara asinkron dengan menentukan sebagai jenis pemanggilan Lambda. 'Event ' Ini dilakukan sebagai berikut:

Konfigurasikan pemanggilan asinkron Lambda di konsol API Gateway

Agar semua pemanggilan menjadi asinkron:

- Dalam permintaan Integrasi, tambahkan X-Amz-Invocation-Type header dengan nilai statis 'Event '.

Bagi klien untuk memutuskan apakah pemanggilan asinkron atau sinkron:

1. Dalam permintaan Metode, tambahkan InvocationType header.
2. Dalam permintaan Integrasi tambahkan X-Amz-Invocation-Type header dengan ekspresi pemetaan. `method.request.header.InvocationType`
3. Klien dapat menyertakan InvocationType: Event header dalam permintaan API untuk pemanggilan asinkron atau untuk pemanggilan sinkron. InvocationType: RequestResponse

Konfigurasikan pemanggilan asinkron Lambda menggunakan OpenAPI

Agar semua pemanggilan menjadi asinkron:

- Tambahkan X-Amz-Invocation-Type header ke x-amazon-apigateway-integration bagian.

```
"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"
}
```

Bagi klien untuk memutuskan apakah pemanggilan asinkron atau sinkron:

1. Tambahkan header berikut pada [OpenAPI Path Item](#) Object.

```
"parameters" : [ {
  "name" : "InvocationType",
  "in" : "header",
  "schema" : {
    "type" : "string"
  }
} ]
```

2. Tambahkan X-Amz-Invocation-Type header ke x-amazon-apigateway-integration bagian.

```
"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. Klien dapat menyertakan InvocationType: Event header dalam permintaan API untuk pemanggilan asinkron atau untuk pemanggilan sinkron. InvocationType: RequestResponse

Konfigurasi pemanggilan asinkron Lambda menggunakan CloudFormation

CloudFormation Template berikut menunjukkan cara mengkonfigurasi `AWS::ApiGateway::Method` untuk pemanggilan asinkron.

Agar semua pemanggilan menjadi asinkron:

```
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'
```

Bagi klien untuk memutuskan apakah pemanggilan asinkron atau sinkron:

```
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'

```

Klien dapat menyertakan `InvocationType`: Event header dalam permintaan API untuk pemanggilan asinkron atau untuk pemanggilan sinkron. `InvocationType`: `RequestResponse`

Menangani kesalahan Lambda di API Gateway

Untuk integrasi kustom Lambda, Anda harus memetakan kesalahan yang dikembalikan oleh Lambda dalam respons integrasi terhadap respons kesalahan HTTP standar untuk klien Anda. Jika tidak, kesalahan Lambda dikembalikan sebagai 200 OK respons secara default dan hasilnya tidak intuitif untuk pengguna API Anda.

Ada dua jenis kesalahan yang dapat dikembalikan Lambda: kesalahan standar dan kesalahan khusus. Di API Anda, Anda harus menangani ini secara berbeda.

Dengan integrasi proxy Lambda, Lambda diperlukan untuk mengembalikan output dari format berikut:

```

{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}

```

Dalam output ini, `statusCode` biasanya 4XX untuk kesalahan klien dan 5XX untuk kesalahan server. API Gateway menangani kesalahan ini dengan memetakan kesalahan Lambda ke respons kesalahan HTTP, sesuai dengan yang ditentukan. `statusCode` Agar API Gateway dapat meneruskan jenis kesalahan (misalnya, `InvalidParameterException`), sebagai bagian dari respons terhadap klien, fungsi Lambda harus menyertakan header (misalnya, `"X-Amzn-ErrorType": "InvalidParameterException"`) di `headers` properti.

Topik

- [Menangani kesalahan Lambda standar di API Gateway](#)
- [Menangani kesalahan Lambda khusus di API Gateway](#)

Menangani kesalahan Lambda standar di API Gateway

AWS Lambda Kesalahan standar memiliki format berikut:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Di sini, `errorMessage` adalah ekspresi string dari kesalahan. `errorType` ini adalah kesalahan atau tipe pengecualian yang bergantung pada bahasa. `stackTrace` ini adalah daftar ekspresi string yang menunjukkan jejak tumpukan yang mengarah ke terjadinya kesalahan.

Misalnya, perhatikan fungsi Lambda berikut JavaScript (Node.js).

```
export const handler = function(event, context, callback) {
  callback(new Error("Malformed input ..."));
};
```

Fungsi ini mengembalikan kesalahan Lambda standar berikut, yang berisi `Malformed input ...` sebagai pesan kesalahan:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Demikian pula, pertimbangkan fungsi Lambda Python berikut, yang memunculkan pesan kesalahan `Exception` yang sama. `Malformed input ...`

```
def lambda_handler(event, context):  
    raise Exception('Malformed input ...')
```

Fungsi ini mengembalikan kesalahan Lambda standar berikut:

```
{  
  "stackTrace": [  
    [  
      "/var/task/lambda_function.py",  
      3,  
      "lambda_handler",  
      "raise Exception('Malformed input ...')"  
    ]  
  ],  
  "errorType": "Exception",  
  "errorMessage": "Malformed input ..."  
}
```

Perhatikan bahwa nilai `errorType` dan `stackTrace` properti bergantung pada bahasa. Kesalahan standar juga berlaku untuk objek kesalahan apa pun yang merupakan perpanjangan dari `Error` objek atau subkelas `Exception` kelas.

Untuk memetakan kesalahan Lambda standar ke respons metode, Anda harus terlebih dahulu memutuskan kode status HTTP untuk kesalahan Lambda yang diberikan. Anda kemudian menetapkan pola ekspresi reguler pada [selectionPattern](#) properti yang [IntegrationResponse](#) terkait dengan kode status HTTP yang diberikan. Di konsol API Gateway, ini `selectionPattern` dilambangkan sebagai regex kesalahan Lambda di bagian Respons Integrasi, di bawah setiap respons integrasi.

Note

API Gateway menggunakan regex gaya pola Java untuk pemetaan respons. Untuk informasi selengkapnya, lihat [Pola](#) dalam Oracle dokumentasi.

Misalnya, gunakan yang berikut ini [put-integration-response](#) untuk menyiapkan `selectionPattern` ekspresi baru:

```
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
```

Pastikan bahwa Anda juga mengatur kode kesalahan yang sesuai (400) pada [respon metode](#). Jika tidak, API Gateway akan menampilkan respons kesalahan konfigurasi yang tidak valid saat runtime.

Note

Saat runtime, API Gateway mencocokkan kesalahan `errorMessage` Lambda dengan pola ekspresi reguler pada `selectionPattern` properti. Jika ada kecocokan, API Gateway mengembalikan kesalahan Lambda sebagai respons HTTP dari kode status HTTP yang sesuai. Jika tidak ada kecocokan, API Gateway mengembalikan kesalahan sebagai respons default atau melempar pengecualian konfigurasi yang tidak valid jika tidak ada respons default yang dikonfigurasi.

Menyetel `selectionPattern` nilai `.*` untuk respons yang diberikan berarti mengatur ulang respons ini sebagai respons default. Ini karena pola pemilihan seperti itu akan cocok dengan semua pesan kesalahan, termasuk null, yaitu, pesan kesalahan yang tidak ditentukan.

Pemetaan yang dihasilkan mengesampingkan pemetaan default. Jika Anda menggunakan `.` + sebagai pola pemilihan untuk memfilter respons, itu mungkin tidak cocok dengan respons yang mengandung ketahuilah bahwa itu mungkin tidak cocok dengan respons yang berisi karakter baris baru (`"\n"`).

Untuk memperbarui `selectionPattern` nilai yang ada menggunakan AWS CLI, panggil [update-integration-response](#) operasi untuk mengganti nilai `/selectionPattern` jalur dengan ekspresi regex yang ditentukan dari pola `Malformed*`

Untuk mengatur `selectionPattern` ekspresi menggunakan konsol API Gateway, masukkan ekspresi di kotak teks regex kesalahan Lambda saat menyiapkan atau memperbarui respons integrasi kode status HTTP tertentu.

Menangani kesalahan Lambda khusus di API Gateway

Alih-alih kesalahan standar yang dijelaskan di bagian sebelumnya, AWS Lambda memungkinkan Anda mengembalikan objek kesalahan kustom sebagai string JSON. Kesalahan dapat berupa objek JSON yang valid. Misalnya, fungsi Lambda berikut JavaScript (Node.js) mengembalikan kesalahan kustom:

```
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));
};
```

Anda harus mengubah `myErrorObj` objek menjadi string JSON sebelum memanggil `callback` untuk keluar dari fungsi. Jika tidak, `myErrorObj` dikembalikan sebagai string dari "[object Object]". Saat metode API Anda terintegrasi dengan fungsi Lambda sebelumnya, API Gateway menerima respons integrasi dengan muatan berikut:

```
{
  "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
  \"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
  \"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
```

Seperti halnya respons integrasi apa pun, Anda dapat melewati respons kesalahan ini apa adanya terhadap respons metode. Atau Anda dapat memiliki template pemetaan untuk mengubah payload menjadi format yang berbeda. Misalnya, pertimbangkan template pemetaan tubuh berikut untuk respons metode kode status 500:

```
{
  errorMessage: $input.path('$.errorMessage');
}
```

Template ini menerjemahkan badan respons integrasi yang berisi string JSON kesalahan kustom ke badan respons metode berikut. Badan respons metode ini berisi objek JSON kesalahan kustom:

```
{
```

```

    "errorMessage" : {
      errorType : "InternalServerError",
      httpStatus : 500,
      requestId : context.awsRequestId,
      trace : {
        "function": "abc()",
        "line": 123,
        "file": "abc.js"
      }
    }
  }
};

```

Bergantung pada persyaratan API Anda, Anda mungkin perlu meneruskan beberapa atau semua properti kesalahan kustom sebagai parameter header respons metode. Anda dapat mencapai ini dengan menerapkan pemetaan kesalahan kustom dari badan respons integrasi ke header respons metode.

Misalnya, ekstensi OpenAPI berikut mendefinisikan pemetaan dari `errorMessage.errorType`,, dan `errorMessage.trace` properti ke `errorMessage.httpStatus`, `errorMessage.trace.function`,, dan `error_type` header `error_status`, `error_trace`, `error_function`, masing-masing. `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"
      },
      ...
    }
  }
}

```

Saat runtime, API Gateway melakukan deserialisasi `integration.response.body` parameter saat melakukan pemetaan header. Namun, deserialisasi ini hanya berlaku untuk body-to-header pemetaan untuk respons kesalahan khusus Lambda dan tidak berlaku untuk pemetaan yang menggunakan `body-to-body $input.body`. Dengan pemetaan `custom-error-body-to-header` ini, klien menerima header berikut sebagai bagian dari respons metode, asalkan, `error_status`, `error_trace`, `error_trace_function`, dan `error_type` header dideklarasikan dalam permintaan metode.

```
"error_status": "500",  
"error_trace": "{\"function\": \"abc()\", \"line\": 123, \"file\": \"abc.js\"}\",  
"error_trace_function": "abc()",  
"error_type": "InternalServerError"
```

`errorMessage.trace` properti badan respons integrasi adalah properti yang kompleks. Hal ini dipetakan ke `error_trace` header sebagai string JSON.

Integrasi HTTP untuk REST APIs di API Gateway

Anda dapat mengintegrasikan metode API dengan titik akhir HTTP menggunakan integrasi proxy HTTP atau integrasi kustom HTTP.

API Gateway mendukung port endpoint berikut: 80, 443 dan 1024-65535.

Dengan integrasi proxy, pengaturannya sederhana. Anda hanya perlu mengatur metode HTTP dan URI titik akhir HTTP, sesuai dengan persyaratan backend, jika Anda tidak peduli dengan pengkodean konten atau caching.

Dengan integrasi khusus, pengaturan lebih terlibat. Selain langkah-langkah penyiapan integrasi proxy, Anda perlu menentukan bagaimana data permintaan masuk dipetakan ke permintaan integrasi dan bagaimana data respons integrasi yang dihasilkan dipetakan ke respons metode.

Topik

- [Siapkan integrasi proxy HTTP di API Gateway](#)
- [Siapkan integrasi kustom HTTP di API Gateway](#)

Siapkan integrasi proxy HTTP di API Gateway

Untuk menyiapkan sumber daya proxy dengan tipe integrasi proxy HTTP, buat sumber daya API dengan parameter jalur serakah (misalnya, `/parent/{proxy+}`) dan integrasikan sumber daya ini

dengan titik akhir backend HTTP (misalnya, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) pada metode. ANY Parameter jalur serakah harus berada di ujung jalur sumber daya.

Seperti sumber daya non-proxy, Anda dapat menyiapkan sumber daya proxy dengan integrasi proxy HTTP menggunakan konsol API Gateway, mengimpor file definisi OpenAPI, atau memanggil API Gateway REST API secara langsung. Untuk petunjuk mendetail tentang penggunaan konsol API Gateway guna mengonfigurasi sumber daya proxy dengan integrasi HTTP, lihat [Tutorial: Membuat REST API dengan integrasi proxy HTTP](#).

File definisi OpenAPI berikut menunjukkan contoh API dengan sumber daya proxy yang terintegrasi dengan situs web. [PetStore](#)

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"
    }
  }
}
}
}

```

Dalam contoh ini, kunci cache dideklarasikan pada parameter `method.request.path.proxy` jalur sumber daya proxy. Ini adalah pengaturan default saat Anda membuat API menggunakan konsol API Gateway. Jalur dasar API (`/test`, sesuai dengan tahap) dipetakan ke PetStore halaman situs web (`/petstore`). Permintaan integrasi tunggal mencerminkan seluruh PetStore situs web menggunakan variabel jalur serakah API dan metode `catch-allANY`. Contoh berikut menggambarkan pencerminan ini.

- Tetapkan **ANY** sebagai **GET** dan **{proxy+}** sebagai **pets**

Permintaan metode dimulai dari frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
```

Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
```

Instance run-time dari ANY metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan 200 OK respons dengan muatan yang berisi batch pertama hewan peliharaan, seperti yang dikembalikan dari backend.

- Tetapkan **ANY** sebagai **GET** dan **{proxy+}** sebagai **pets?type=dog**

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
HTTP/1.1
```

Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
```

Instance run-time dari ANY metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan 200 OK respons dengan payload yang berisi batch pertama dog tertentu, seperti yang dikembalikan dari backend.

- Tetapkan **ANY** sebagai **GET** dan **{proxy+}** sebagai **pets/{petId}**

Permintaan metode dimulai dari frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
```

Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Instance run-time dari ANY metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan 200 OK respons dengan muatan yang berisi hewan peliharaan yang ditentukan, seperti yang dikembalikan dari backend.

- Tetapkan **ANY** sebagai **POST** dan **{proxy+}** sebagai **pets**

Permintaan metode dimulai dari frontend:

```
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
}
```

Permintaan integrasi dikirim ke backend:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

Instance run-time dari ANY metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan 200 OK respons dengan muatan yang berisi hewan peliharaan yang baru dibuat, seperti yang dikembalikan dari backend.

- Tetapkan **ANY** sebagai **GET** dan **{proxy+}** sebagai **pets/cat**

Permintaan metode dimulai dari frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

Instance run-time dari jalur sumber daya proxy tidak sesuai dengan titik akhir backend dan permintaan yang dihasilkan tidak valid. Akibatnya, 400 Bad Request respons dikembalikan dengan pesan kesalahan berikut.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    },
    {
      "key": "Pet2.price",
      "message": "Missing required field"
    }
  ]
}
```

- Tetapkan **ANY** sebagai **GET** dan **{proxy+}** sebagai **null**

Permintaan metode dimulai dari frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
```

Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Sumber daya yang ditargetkan adalah induk dari sumber daya proxy, tetapi instance run-time dari ANY metode ini tidak ditentukan dalam API pada sumber daya tersebut. Akibatnya, GET permintaan ini mengembalikan 403 Forbidden respons dengan pesan Missing Authentication Token kesalahan seperti yang dikembalikan oleh API Gateway. Jika API mengekspos GET metode ANY or pada sumber daya induk (/), panggilan akan menampilkan 404 Not Found respons dengan Cannot GET /petstore pesan yang dikembalikan dari backend.

Untuk permintaan klien apa pun, jika URL titik akhir yang ditargetkan tidak valid atau kata kerja HTTP valid tetapi tidak didukung, backend mengembalikan respons 404 Not Found Untuk metode HTTP yang tidak didukung, 403 Forbidden respons dikembalikan.

Siapkan integrasi kustom HTTP di API Gateway

Dengan integrasi kustom HTTP, juga dikenal sebagai integrasi non-proxy, Anda memiliki kontrol lebih besar atas data mana yang harus dilewatkan antara metode API dan integrasi API dan cara meneruskan data. Anda melakukan ini menggunakan pemetaan data.

[Sebagai bagian dari penyiapan permintaan metode, Anda menetapkan properti `requestParameters` pada sumber daya Metode.](#) Ini menyatakan parameter permintaan metode mana, yang disediakan dari klien, yang akan dipetakan ke parameter permintaan integrasi atau properti badan yang berlaku sebelum dikirim ke backend. Kemudian, sebagai bagian dari penyiapan permintaan integrasi, Anda menetapkan properti [requestParameters](#) pada sumber daya Integrasi [yang sesuai](#) untuk menentukan pemetaan. parameter-to-parameter Anda juga mengatur properti [requestTemplates](#) untuk menentukan templat pemetaan, satu untuk setiap jenis konten yang didukung. Metode pemetaan template memetakan parameter permintaan, atau isi, ke badan permintaan integrasi.

Demikian pula, sebagai bagian dari pengaturan respons metode, Anda mengatur properti [ResponseParameters](#) pada sumber daya. [MethodResponse](#) Ini menyatakan parameter respons metode mana, yang akan dikirim ke klien, yang akan dipetakan dari parameter respons integrasi atau properti tubuh tertentu yang berlaku yang dikembalikan dari backend. Anda kemudian mengatur [SelectionPattern](#) untuk memilih respons integrasi berdasarkan respons dari backend. Untuk integrasi HTTP non-proxy, ini adalah ekspresi reguler. Misalnya, untuk memetakan semua kode status respons HTTP 2xx dari titik akhir HTTP ke pemetaan keluaran ini, gunakan `2\d{2}`

Note

API Gateway menggunakan regex gaya pola Java untuk pemetaan respons. Untuk informasi selengkapnya, lihat [Pola](#) dalam Oracle dokumentasi.

Kemudian, sebagai bagian dari pengaturan respons integrasi, Anda mengatur properti [ResponseParameters](#) pada sumber daya yang [IntegrationResponses](#)sesuai untuk menentukan pemetaan. parameter-to-parameter Anda juga mengatur peta [ResponseTemplates](#) untuk menentukan templat pemetaan, satu untuk setiap jenis konten yang didukung. Template pemetaan memetakan parameter respons integrasi, atau properti badan respons integrasi, ke badan respons metode.

Untuk informasi selengkapnya tentang menyiapkan templat pemetaan, lihat [Transformasi data untuk REST API di API Gateway](#).

Streaming respons integrasi untuk integrasi proxy Anda di API Gateway

Anda dapat mengonfigurasi integrasi proksi untuk mengontrol cara API Gateway menghasilkan respons integrasi Anda. Secara default, API Gateway menunggu untuk menerima respons lengkap sebelum memulai transmisi. Namun, jika Anda menyetel mode transfer respons integrasi ke `STREAM`, API Gateway tidak menunggu respons dihitung sepenuhnya sebelum mengirimkannya ke klien. Aliran respons berfungsi untuk semua jenis titik akhir API REST.

Gunakan streaming respons untuk kasus penggunaan berikut:

- Turunkan time-to-first-byte (TTFB) untuk aplikasi AI generatif seperti chatbots.
- Streaming file gambar, video, atau musik besar tanpa menggunakan URL S3 yang telah ditandatangani sebelumnya.
- Lakukan operasi yang berjalan lama sambil melaporkan kemajuan tambahan seperti peristiwa terkirim server (SSE).
- Melebihi batas muatan respons 10 MB API Gateway.
- Melebihi batas waktu tunggu 29 detik API Gateway tanpa meminta peningkatan batas waktu integrasi.
- Menerima payload biner tanpa mengkonfigurasi jenis media biner.

Pertimbangan untuk streaming payload respons

Pertimbangan berikut dapat memengaruhi penggunaan streaming payload respons Anda:

- Anda hanya dapat menggunakan streaming payload respons untuk `HTTP_PROXY` atau jenis `AWS_PROXY` integrasi. Ini termasuk integrasi proxy Lambda dan integrasi pribadi yang menggunakan integrasi `HTTP_PROXY`.
- Pengaturan mode transfer default adalah `BUFFERED`. Untuk menggunakan streaming respons, Anda harus mengubah mode transfer respons ke `STREAM`.
- Streaming respons hanya didukung untuk REST APIs.
- Permintaan streaming tidak didukung.
- Anda dapat melakukan streaming respons Anda hingga 15 menit.
- Streaming Anda tunduk pada batas waktu idle. Untuk titik akhir Regional atau pribadi, batas waktu adalah 5 menit. Untuk titik akhir yang dioptimalkan tepi, batas waktu adalah 30 detik.
- Jika Anda menggunakan streaming respons untuk REST API Regional dengan CloudFront distribusi Anda sendiri, Anda dapat mencapai waktu idle lebih dari 30 detik dengan meningkatkan

batas waktu respons distribusi Anda CloudFront. Untuk informasi selengkapnya, lihat [Batas waktu respons](#).

- Saat mode transfer respons disetel ke `STREAM`, API Gateway tidak dapat mendukung fitur yang memerlukan buffering seluruh respons integrasi. Karena itu, fitur berikut tidak didukung dengan streaming respons:
 - Caching titik akhir
 - Pengkodean konten. Jika Anda ingin mengompres respons integrasi Anda, lakukan ini dalam integrasi Anda.
 - Transformasi respons dengan VTL
- Dalam setiap respons streaming, muatan respons 10MB pertama tidak tunduk pada batasan bandwidth apa pun. Data payload respons melebihi 10MB dibatasi hingga 2MB/s.
- Ketika koneksi antara klien dan API Gateway atau antara API Gateway dan Lambda ditutup karena batas waktu, fungsi Lambda mungkin terus dijalankan. Untuk informasi selengkapnya, lihat [Mengonfigurasi batas waktu fungsi Lambda](#).
- Streaming respons menimbulkan biaya. Untuk informasi selengkapnya, lihat [Harga API Gateway](#).

Siapkan integrasi proxy HTTP dengan streaming respons payload di API Gateway

Saat mengatur streaming payload respons, Anda menentukan mode transfer respons dalam permintaan integrasi metode Anda. Anda mengonfigurasi setelan ini dalam permintaan integrasi untuk mengontrol perilaku API Gateway sebelum dan selama respons integrasi. Saat Anda menggunakan streaming respons, Anda dapat mengonfigurasi batas waktu integrasi hingga 15 menit.

Saat Anda menggunakan streaming respons payload dengan `HTTP_PROXY` integrasi, API Gateway tidak akan mengirim kode status respons HTTP atau header respons HTTP apa pun hingga sepenuhnya menerima semua header.

Buat integrasi proxy HTTP dengan streaming respons payload

Prosedur berikut menunjukkan cara mengimpor API baru dengan `responseTransferMode` set ke `STREAM`. Jika Anda memiliki API integrasi yang ada dan ingin memodifikasinya `responseTransferMode`, lihat [the section called “Perbarui mode transfer respons untuk integrasi proxy HTTP”](#).

Konsol Manajemen AWS

Untuk membuat integrasi proxy HTTP dengan streaming respons payload

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih Buat sumber daya.
4. Untuk Nama sumber daya, masukkan **streaming**.
5. Pilih Buat sumber daya.
6. Dengan sumber daya /streaming yang dipilih, pilih Create method.
7. Untuk jenis Metode, pilih APAPUN.
8. Untuk jenis Integrasi, pilih HTTP.
9. Pilih integrasi proxy HTTP.
10. Untuk mode transfer Respons, pilih Streaming.
11. Untuk metode HTTP, pilih metode.
12. Untuk URL Endpoint, masukkan titik akhir integrasi. Pastikan Anda memilih titik akhir yang menghasilkan muatan besar untuk dialirkan kembali kepada Anda.
13. Pilih metode Buat.

Setelah Anda membuat metode, terapkan API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

AWS CLI

Untuk membuat API baru dengan streaming respons payload

1. Salin file Open API berikut, lalu simpan sebagai `ResponseStreamDemoSwagger.yaml`. Dalam file ini, `responseTransferMode` diatur ke `STREAM`. Titik akhir integrasi diatur

ke`https://example.com`, tetapi kami menyarankan Anda memodifikasinya ke titik akhir yang menghasilkan muatan besar untuk dialirkan kembali kepada Anda.

```
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"
```

- Gunakan `import-rest-api` perintah berikut untuk mengimpor definisi OpenAPI Anda:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

- Gunakan `create-deployment` perintah berikut untuk menerapkan API baru Anda ke panggung:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c3 \
  --stage-name prod \
  --region us-west-1
```

Perbarui mode transfer respons untuk integrasi proxy HTTP

Prosedur berikut menunjukkan cara memperbarui mode transfer respons untuk integrasi proxy HTTP.

Konsol Manajemen AWS

Untuk memperbarui mode transfer respons untuk integrasi proxy HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih metode.
4. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
5. Untuk mode transfer Respons, pilih Streaming.
6. Pilih Simpan.

Setelah memperbarui metode, terapkan API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

AWS CLI

update-integrationPerintah berikut memperbarui mode transfer integrasi dari BUFFERED keSTREAM. Untuk yang ada APIs, mode transfer respons untuk semua integrasi diatur keBUFFERED.

```
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
```

Anda perlu menerapkan ulang API agar perubahan diterapkan. Jika Anda menyesuaikan batas waktu integrasi, nilai batas waktu ini akan dihapus, karena API Gateway mengalirkan respons Anda hingga 5 menit.

`update-integration` Perintah berikut memperbarui mode transfer integrasi dari `STREAM` ke `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
```

Anda perlu menerapkan ulang API agar perubahan diterapkan.

Siapkan integrasi proxy Lambda dengan streaming respons payload di API Gateway

Anda dapat melakukan streaming respons fungsi Lambda untuk meningkatkan kinerja time to first byte (TTFB) dan mengirim sebagian tanggapan kembali ke klien saat tersedia. API Gateway mengharuskan Anda menggunakan API [InvokeWithResponseStream](#) Lambda untuk menjalankan fungsi Lambda Anda. API Gateway meneruskan objek peristiwa ke fungsi Lambda. Fungsi backend Lambda mem-parsing data permintaan yang masuk untuk menentukan respons yang dikembalikan. Agar API Gateway dapat melakukan streaming output Lambda, fungsi Lambda harus menampilkan [format](#) yang diperlukan oleh API Gateway.

Perbedaan integrasi proxy Lambda antara mode transfer respons streaming dan buffer

Daftar berikut menjelaskan perbedaan antara integrasi proxy Lambda dan integrasi proxy Lambda untuk streaming respons:

- API Gateway menggunakan [InvokeWithResponseStream](#) API untuk menjalankan integrasi proxy Lambda untuk streaming respons. Ini menghasilkan URI yang berbeda, yaitu sebagai berikut:

```
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
```

ARN ini menggunakan tanggal yang berbeda untuk versi API dan tindakan layanan yang berbeda dibandingkan dengan integrasi proxy Lambda.

Jika Anda menggunakan konsol API Gateway untuk streaming respons, konsol menggunakan URI yang benar untuk Anda.

- Dalam integrasi proxy Lambda, API Gateway mengirimkan respons ke klien hanya setelah menerima respons penuh dari Lambda. Dalam integrasi proxy Lambda untuk streaming respons, API Gateway memulai aliran muatan setelah menerima metadata dan pembatas yang valid dari Lambda.
- Integrasi proxy Lambda untuk streaming respons menggunakan format input yang sama dengan integrasi proxy, tetapi memerlukan format output yang berbeda.

Format integrasi proxy Lambda untuk streaming respons

Saat API Gateway memanggil fungsi Lambda dengan streaming respons, format input sama dengan format input fungsi Lambda untuk integrasi proxy. Untuk informasi selengkapnya, lihat [the section called “Format input fungsi Lambda untuk integrasi proxy”](#).

Saat Lambda mengalirkan respons ke API Gateway, respons harus mematuhi format berikut. Format ini menggunakan pembatas untuk memisahkan metadata JSON dan muatan mentah. Dalam hal ini, data payload dialirkan saat ditransmisikan oleh fungsi Lambda streaming Anda:

```
{
  "headers": {"headerName": "headerValue", ...},
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "cookies" : ["cookie1", "cookie2"],
  "statusCode": httpStatusCode
}<DELIMITER>PAYLOAD1 | PAYLOAD2 | PAYLOAD3
```

Dalam output:

- `statusCode` Tombol `headers` `multiValueHeaders`, `cookies`, dan dapat tidak ditentukan jika tidak ada header respons tambahan yang akan dikembalikan.
- `headers` Kuncinya hanya dapat berisi header nilai tunggal.
- Output mengharapkan header berisi salah satu atau `Transfer-Encoding: chunked`. `Content-length: number` Jika fungsi Anda tidak mengembalikan salah satu header ini, API Gateway akan ditambahkan `Transfer-Encoding: chunked` ke header respons.
- `multiValueHeaders` Kuncinya dapat berisi header multi-nilai serta header nilai tunggal. Anda dapat menggunakan `multiValueHeaders` kunci untuk menentukan semua header tambahan Anda, termasuk yang bernilai tunggal.

- Jika Anda menentukan nilai untuk keduanya `headers` dan `multiValueHeaders`, API Gateway menggabungkannya ke dalam satu daftar. Jika pasangan kunci-nilai yang sama ditentukan di keduanya, hanya nilai dari yang `multiValueHeaders` akan muncul dalam daftar gabungan.
- Metadata harus JSON yang valid. Hanya `headers`, `multiValueHeaders`, `cookies` dan `statusCode` kuncinya didukung.
- Anda harus memberikan pembatas setelah metadata JSON. Pembatas harus 8 byte nol dan harus muncul dalam 16KB pertama data aliran.
- API Gateway tidak memerlukan format khusus untuk payload respons metode.

Jika Anda menggunakan URL fungsi untuk melakukan streaming fungsi Lambda Anda, Anda harus memodifikasi input dan output fungsi Lambda Anda untuk memenuhi persyaratan ini.

Jika output fungsi Lambda Anda tidak mematuhi persyaratan format ini, API Gateway mungkin masih menjalankan fungsi Lambda Anda. Tabel berikut menunjukkan kombinasi setelan permintaan integrasi API dan kode fungsi Lambda yang didukung oleh API Gateway. Ini termasuk kombinasi yang didukung untuk mode transfer respons buffer.

Mode transfer respons	Kode fungsi mematuhi format yang diperlukan	Lambda memanggil API	Didukung oleh API Gateway
Streaming	Ya	InvokeWithResponseStream	Ya. API Gateway mengalirkan respons Anda.
Streaming	Tidak	InvokeWithResponseStream	Tidak. API Gateway memanggil fungsi Lambda Anda dan mengembalikan respons kesalahan 500.
Streaming	Ya	Memohon	Tidak. API Gateway tidak mendukung konfigurasi integrasi ini.

Mode transfer respons	Kode fungsi mematuhi format yang diperlukan	Lambda memanggil API	Didukung oleh API Gateway
Streaming	Tidak	Memohon	Tidak. API Gateway tidak mendukung konfigurasi integrasi ini.
Buffer	Ya	InvokeWithResponseStream	Tidak. API Gateway tidak mendukung konfigurasi integrasi ini.
Buffer	Tidak	InvokeWithResponseStream	Tidak. API Gateway tidak mendukung konfigurasi integrasi ini.
Buffer	Ya	Memohon	API Gateway mengembalikan header HTTP dan kode status tetapi bukan badan respons.
Buffer	Tidak	Memohon	Ya. Ini adalah integrasi proxy Lambda. Untuk informasi selengkapnya, lihat Integrasi proxy Lambda .

Konfigurasi integrasi proxy Lambda dengan streaming respons payload di API Gateway

Saat mengatur streaming payload respons, Anda menentukan mode transfer dalam permintaan integrasi sumber daya Anda. Anda mengonfigurasi setelan ini dalam permintaan integrasi untuk mengontrol perilaku API Gateway sebelum dan selama respons integrasi.

Contoh fungsi Lambda untuk streaming respons

Fungsi Lambda Anda harus mematuhi. [the section called “Format integrasi proxy Lambda untuk streaming respons”](#) Kami menyarankan Anda menggunakan salah satu dari tiga contoh fungsi Lambda untuk menguji streaming respons. Saat Anda membuat fungsi Lambda, pastikan untuk melakukan hal berikut:

- Berikan batas waktu yang memadai untuk fungsi Anda. Kami menyarankan Anda mengonfigurasi batas waktu minimal 1 menit untuk mempelajari tentang streaming respons. Saat Anda membuat sumber daya produksi, pastikan batas waktu fungsi Lambda Anda mencakup siklus permintaan penuh. Untuk informasi selengkapnya, lihat [Mengonfigurasi batas waktu fungsi Lambda](#).
- Gunakan runtime Node.js terbaru.
- Gunakan Wilayah tempat streaming respons Lambda tersedia.

Using `HttpResponseStream.from`

Contoh kode berikut mengalirkan objek metadata JSON dan muatan kembali ke klien menggunakan `awslambda.HttpResponseStream()` metode tanpa menggunakan metode pipeline. Anda tidak perlu membuat pembatas. Untuk informasi selengkapnya, lihat [Menulis fungsi Lambda yang mendukung streaming respons](#).

```
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 merekomendasikan bahwa ketika Anda menulis fungsi yang mendukung streaming respons, Anda menggunakan `awslambda.streamifyResponse()` dekorator yang disediakan runtime Node.js asli, dan metodenya. `pipeline()` Saat Anda menggunakan metode pipeline, Anda tidak perlu membuat pembatas, Lambda melakukan ini untuk Anda. Untuk informasi selengkapnya, lihat [Menulis fungsi Lambda yang mendukung streaming respons](#).

Contoh kode berikut mengalirkan objek metadata JSON dan tiga muatan kembali ke klien.

```
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

Contoh kode berikut mengalirkan objek metadata JSON dan tiga muatan kembali ke klien tanpa menggunakan metode ini. `awslambda.HttpResponseStream()` Tanpa `awslambda.HttpResponseStream()` metode ini, Anda harus menyertakan pembatas 8 byte nol antara metadata dan muatan.

```
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();
});
```

Buat integrasi proxy Lambda dengan streaming respons payload

Prosedur berikut menunjukkan cara membuat integrasi proxy Lambda dengan streaming respons payload. Gunakan contoh fungsi Lambda atau buat sendiri.

Konsol Manajemen AWS

Untuk membuat integrasi proxy Lambda dengan streaming respons payload

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih Buat sumber daya.
4. Untuk Nama sumber daya, masukkan **streaming**.
5. Pilih Buat sumber daya.
6. Dengan sumber daya /streaming yang dipilih, pilih Create method.

7. Untuk jenis Metode, pilih APAPUN.
8. Untuk jenis Integrasi, pilih Lambda.
9. Pilih integrasi proxy Lambda.
10. Untuk mode transfer Respons, pilih Streaming.
11. Untuk fungsi Lambda, pilih nama fungsi Lambda Anda.

Konsol API Gateway secara otomatis menggunakan [InvokeWithResponseStream](#) API untuk menjalankan fungsi Lambda. Anda bertanggung jawab untuk menulis fungsi Lambda yang mendukung streaming respons. Sebagai contoh, lihat [the section called “Contoh fungsi Lambda untuk streaming respons”](#).

12. Pilih metode Buat.

Setelah Anda membuat metode, terapkan API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

AWS CLI

Prosedur berikut menunjukkan cara mengimpor API baru dengan `responseTransferMode` set ke `STREAM`. Jika Anda memiliki API integrasi yang ada dan ingin memodifikasinya `responseTransferMode`, lihat [the section called “Perbarui mode transfer respons untuk integrasi proxy Lambda”](#).

Untuk membuat API baru dengan streaming respons payload

1. Salin file Open API berikut, lalu simpan sebagai `fileResponseStreamDemoSwagger.yaml`. Dalam file ini, `responseTransferMode` diatur ke `STREAM`, dan URI integrasi diatur ke `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`.

Ganti nama fungsi dari `my-function` dengan fungsi berkemampuan streaming dan ganti kredensialnya dengan peran IAM yang memiliki kebijakan yang memungkinkan layanan `apigateway` memanggil fungsi Lambda.

```
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"
```

Alih-alih menyediakan peran IAM untuk kredensial, Anda dapat menggunakan perintah untuk Lambda `add-permission` untuk menambahkan izin berbasis sumber daya.

- Gunakan `import-rest-api` perintah berikut untuk mengimpor definisi OpenAPI Anda:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

- Gunakan `create-deployment` perintah berikut untuk menerapkan API baru Anda ke panggung:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c2 \
```

```
--stage-name prod \  
--region us-west-1
```

Perbarui mode transfer respons untuk integrasi proxy Lambda

Prosedur berikut menunjukkan cara memperbarui mode transfer respons untuk integrasi proxy Lambda. Saat Anda mengubah mode transfer respons ke streaming, perbarui fungsi Lambda Anda sehingga mematuhi persyaratan untuk streaming respons. Gunakan contoh fungsi Lambda atau buat sendiri.

Konsol Manajemen AWS

Untuk memperbarui mode transfer respons untuk integrasi proxy Lambda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih metode.
4. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
5. Untuk mode transfer Respons, pilih Streaming.
6. Untuk fungsi Lambda, pilih nama fungsi Lambda Anda.
7. Pilih Simpan.

Setelah memperbarui metode, terapkan API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan **prod**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

AWS CLI

1. Perbarui fungsi Lambda Anda agar diaktifkan streaming.

- Gunakan AWS CLI perintah berikut untuk memperbarui URI integrasi dan mode transfer respons integrasi Anda:

```
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
```

- Menerapkan ulang API Anda agar perubahan diterapkan.

Memecahkan masalah dengan streaming respons di API Gateway

Panduan pemecahan masalah berikut dapat membantu menyelesaikan masalah dengan API yang menggunakan streaming respons.

Pemecahan masalah umum

Anda dapat menggunakan tab pengujian [TestInvokeMethod](#) atau konsol untuk menguji respons streaming Anda. Pertimbangan berikut dapat memengaruhi penggunaan pemanggilan pengujian Anda untuk streaming respons:

- Saat Anda menguji metode Anda, API Gateway menyangga payload respons yang dialirkan. Setelah salah satu kondisi berikut terpenuhi, API Gateway mengembalikan respons satu kali yang berisi muatan buffer:
 - Permintaan selesai
 - 35 detik telah berlalu
 - Lebih dari 1 MB payload respons telah di-buffer
- Jika lebih dari 35 detik berlalu sebelum metode Anda mengembalikan status respons HTTP dan semua header, maka status respons yang dikembalikan adalah 0. `TestInvokeMethod`
- API Gateway tidak menghasilkan log eksekusi apa pun.

Setelah menerapkan API, Anda dapat menguji respons streaming dengan menggunakan perintah curl. Kami menyarankan Anda menggunakan `-i` opsi untuk menyertakan header respons protokol dalam output. Untuk melihat data respons saat tiba, gunakan opsi curl `--no-buffer`

Memecahkan masalah kesalahan cURL

Jika Anda menguji integrasi dan Anda menerima kesalahan `curl: (18) transfer closed with outstanding read data remaining`, pastikan batas waktu integrasi Anda cukup lama. Jika Anda menggunakan fungsi Lambda, Anda perlu memperbarui batas waktu respons fungsi Lambda. Untuk informasi selengkapnya, lihat [Mengonfigurasi batas waktu fungsi Lambda](#).

Memecahkan masalah menggunakan pencatatan akses

Anda dapat menggunakan log akses untuk tahap REST API untuk mencatat dan memecahkan masalah aliran respons Anda. Selain variabel yang ada, Anda dapat menggunakan variabel log akses berikut:

`$context.integration.responseTransferMode`

Mode transfer respons integrasi Anda. Ini bisa salah satu `BUFFERED` atau `STREAMED`.

`$context.integration.timeToAllHeaders`

Waktu antara saat API Gateway menetapkan koneksi integrasi saat menerima semua header respons integrasi dari klien.

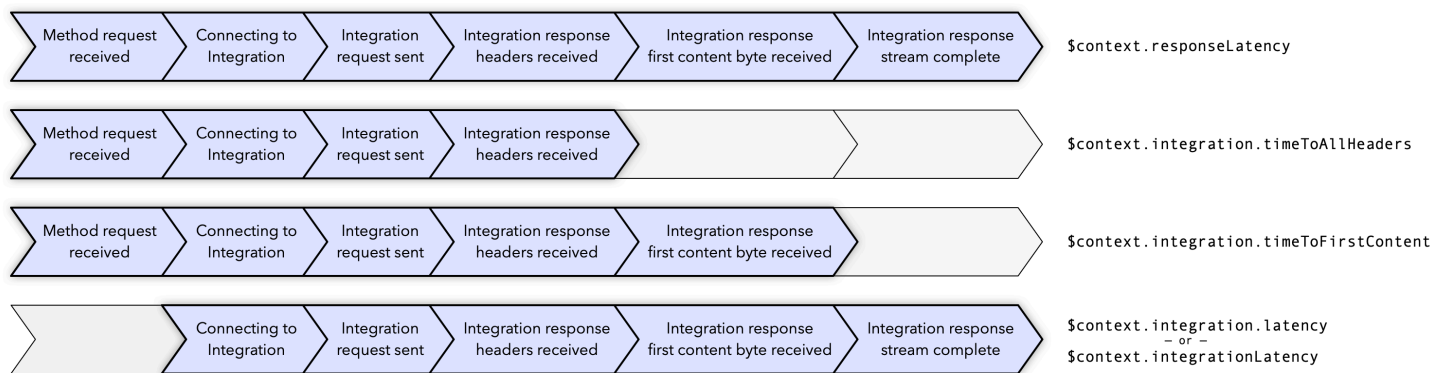
`$context.integration.timeToFirstContent`

Waktu antara saat API Gateway menetapkan koneksi integrasi saat menerima byte konten pertama.

`$context.integration.latency` atau `$context.integrationLatency`

Waktu ketika API Gateway menetapkan koneksi integrasi saat aliran respons integrasi selesai.

Gambar berikut menunjukkan bagaimana variabel log akses ini mewakili komponen yang berbeda dari aliran respons.



Untuk informasi selengkapnya tentang log akses, lihat [the section called “CloudWatch log”](#). Anda juga dapat menggunakan X-Ray untuk memantau aliran respons Anda. Lihat informasi yang lebih lengkap di [the section called “X-Ray”](#).

Integrasi pribadi untuk REST APIs di API Gateway

Gunakan integrasi pribadi untuk mengekspos HTTP/HTTPS sumber daya Anda dalam VPC Amazon untuk akses oleh klien di luar VPC. Ini memperluas akses ke sumber daya VPC pribadi Anda di luar batas VPC. Anda dapat mengontrol akses ke API Anda dengan menggunakan salah satu [metode otorisasi](#) yang didukung API Gateway.

Untuk membuat integrasi pribadi, Anda harus terlebih dahulu membuat tautan VPC. API Gateway mendukung tautan VPC V2 untuk REST. APIs VPC link V2 memungkinkan Anda membuat integrasi pribadi yang menghubungkan REST API Anda ke Application Load Balancers tanpa menggunakan Network Load Balancer. Menggunakan Application Load Balancer memungkinkan Anda terhubung ke aplikasi ECSs berbasis container Amazon dan banyak backend lainnya. Tautan VPC V1 dianggap sebagai jenis integrasi lama. Meskipun didukung oleh API Gateway, kami menyarankan Anda untuk tidak membuat tautan VPC baru V1.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan integrasi pribadi Anda:

- Semua sumber daya harus dimiliki oleh yang sama Akun AWS. Ini termasuk penyeimbang beban, tautan VPC, dan REST API.
- Secara default, lalu lintas integrasi pribadi menggunakan protokol HTTP. Untuk menggunakan HTTPS, tentukan [uri](#) yang berisi nama server yang aman, seperti `https://example.com:443/test`.

- Dalam integrasi pribadi, API Gateway menyertakan bagian [tahap](#) titik akhir API dalam permintaan ke sumber daya backend Anda. Misalnya, jika Anda meminta test tahap API, API Gateway menyertakan test/path permintaan integrasi pribadi Anda. Untuk menghapus nama panggung dari permintaan ke sumber daya backend Anda, gunakan [pemetaan parameter](#) untuk membuat penggantian variabel. `$context.requestOverride.path`
- Integrasi pribadi dengan AWS Cloud Map tidak didukung.

Topik

- [Siapkan tautan VPC V2 di API Gateway](#)
- [Siapkan integrasi pribadi](#)
- [Integrasi pribadi menggunakan tautan VPC V1 \(warisan\)](#)

Siapkan tautan VPC V2 di API Gateway

Tautan VPC memungkinkan Anda membuat integrasi pribadi yang menghubungkan rute API Anda ke sumber daya pribadi di VPC, seperti Application Load Balancers atau aplikasi berbasis container Amazon ECS. Integrasi pribadi menggunakan tautan VPC V2 untuk merangkum koneksi antara API Gateway dan sumber daya VPC yang ditargetkan. Anda dapat menggunakan kembali tautan VPC di berbagai sumber daya dan API.

Saat Anda membuat tautan VPC, API Gateway membuat dan mengelola [antarmuka jaringan elastis](#) untuk tautan VPC V2 di akun Anda. Proses ini dapat menghabiskan waktu beberapa menit. Ketika tautan VPC V2 siap digunakan, statusnya bertransisi dari ke. PENDING AVAILABLE

Note

Jika tidak ada lalu lintas yang dikirim melalui tautan VPC selama 60 hari, itu menjadi INACTIVE. Saat tautan VPC dalam INACTIVE status, API Gateway menghapus semua antarmuka jaringan tautan VPC. Hal ini menyebabkan permintaan API yang bergantung pada tautan VPC gagal. Jika permintaan API dilanjutkan, API Gateway akan menyediakan kembali antarmuka jaringan. Diperlukan beberapa menit untuk membuat antarmuka jaringan dan mengaktifkan kembali tautan VPC. Anda dapat menggunakan status tautan VPC untuk memantau status tautan VPC Anda.

Buat tautan VPC V2 dengan menggunakan AWS CLI

Untuk membuat tautan VPC V2, semua sumber daya yang terlibat harus dimiliki oleh akun yang sama AWS . Perintah [create-vpc-link](#) berikut membuat tautan VPC:

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \  
  --subnet-ids subnet-aaaa subnet-bbbb \  
  --security-group-ids sg1234 sg5678
```

Note

Tautan VPC V2 tidak dapat diubah. Setelah membuat tautan VPC V2, Anda tidak dapat mengubah subnet atau grup keamanannya.

Hapus tautan VPC V2 dengan menggunakan AWS CLI

[Perintah delete-vpc-link](#) berikut menghapus tautan VPC.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Ketersediaan berdasarkan Wilayah

Tautan VPC V2 didukung di Wilayah dan Zona Ketersediaan berikut:

Nama wilayah	Region	Zona Ketersediaan yang Didukung
AS Timur (Ohio)	us-east-2	gunakan2-az1, gunakan2-az2, gunakan2-az3
US East (Northern Virginia)	us-east-1	gunakan1-az1, gunakan1-az2, gunakan1-az4, gunakan1-az5, gunakan1-az6
AS Barat (California Utara)	us-west-1	usw1-az1, usw1-az3

Nama wilayah	Region	Zona Ketersediaan yang Didukung
AS Barat (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asia Pasifik (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Asia Pasifik (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia Pacific (Osaka)	ap-northeast-3	apne3-az1, apne3-az2, apne3-az3
Asia Pasifik (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia Pasifik (Singapura)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3
Asia Pasifik (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asia Pasifik (Tokyo)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Kanada (Pusat)	ca-central-1	cac1-az1, cac1-az2
Eropa (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Eropa (Irlandia)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Eropa (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Eropa (Paris)	eu-west-3	euw3-az1, euw3-az3

Nama wilayah	Region	Zona Ketersediaan yang Didukung
Eropa (Spanyol)	eu-south-2	eus2-az1, eus2-az2, eus2-az3
Eropa (Stockholm)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Timur Tengah (Bahrain)	me-south-1	mes1-az1, mes1-az2, mes1-az3
Amerika Selatan (Sao Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Siapkan integrasi pribadi

Untuk membuat integrasi pribadi dengan Application Load Balancer atau Network Load Balancer, Anda membuat integrasi proxy HTTP, menentukan [tautan VPC V2](#) yang akan digunakan, dan menyediakan ARN Network Load Balancer atau Application Load Balancer. Secara default, lalu lintas integrasi pribadi menggunakan protokol HTTP. Untuk menggunakan HTTPS, tentukan [uri](#) yang berisi nama server yang aman, seperti `https://example.com:443/test`. Untuk tutorial lengkap tentang cara membuat REST API dengan integrasi pribadi, lihat [the section called “Tutorial: Buat REST API dengan integrasi pribadi”](#).

Buat integrasi pribadi

Prosedur berikut menunjukkan cara membuat integrasi pribadi yang terhubung ke penyeimbang beban dengan menggunakan tautan VPC V2.

Konsol Manajemen AWS

Untuk tutorial tentang cara membuat integrasi pribadi lihat, [the section called “Tutorial: Buat REST API dengan integrasi pribadi”](#).

AWS CLI

Perintah [put-integration](#) berikut membuat integrasi pribadi yang terhubung ke penyeimbang beban dengan menggunakan tautan VPC V2:

```
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
```

Alih-alih langsung memberikan ID koneksi, Anda dapat menggunakan variabel panggung sebagai gantinya. Saat menerapkan API ke sebuah panggung, Anda menyetel ID VPC link V2. Perintah [put-integration](#) berikut membuat integrasi pribadi menggunakan variabel stage untuk 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.vpcLinkV2Id}"
```

Pastikan untuk mengutip dua kali ekspresi variabel panggung (\$ {stageVariables.vpcLinkV2Id}) dan keluar dari karakter \$.

OpenAPI

Anda dapat menyiapkan API dengan integrasi pribadi dengan mengimpor file OpenAPI API. Pengaturannya mirip dengan definisi OpenAPI API dengan integrasi HTTP, dengan pengecualian berikut:

- Anda harus secara eksplisit mengatur `connectionType` ke `VPC_LINK`
- Anda harus secara eksplisit mengatur `connectionId` ke ID dari `VpcLinkV2` atau ke variabel tahap yang merujuk ID dari sebuah `VpcLinkV2`
- `uriParameter` dalam integrasi pribadi menunjuk ke HTTP/HTTPS titik akhir di VPC, tetapi digunakan sebagai gantinya untuk mengatur header permintaan Host integrasi.
- `uriParameter` dalam integrasi pribadi dengan titik akhir HTTPS di VPC digunakan untuk memverifikasi nama domain yang dinyatakan terhadap yang ada di sertifikat yang diinstal pada titik akhir VPC.

Anda dapat menggunakan variabel tahap untuk mereferensikan `VpcLinkV2` ID. Atau Anda dapat menetapkan nilai ID langsung ke `connectionId`.

File OpenAPI berformat JSON berikut menunjukkan contoh API dengan tautan VPC seperti yang direferensikan oleh variabel tahap (): `${stageVariables.vpcLinkIdV2}`

```
{
  "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"
    }
}
}

```

Perbarui integrasi pribadi

Contoh berikut memperbarui tautan VPC V2 untuk integrasi pribadi.

Konsol Manajemen AWS

Untuk memperbarui integrasi pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API dengan integrasi pribadi.

3. Pilih sumber daya dan metode yang menggunakan integrasi pribadi.
4. Pada tab Permintaan integrasi, di bawah pengaturan permintaan Integrasi, pilih Edit.
5. Anda dapat mengedit pengaturan integrasi pribadi Anda. Jika saat ini Anda menggunakan tautan VPC V1, Anda dapat mengubah tautan VPC Anda ke tautan VPC V2.
6. Pilih Simpan.
7. Menerapkan ulang API Anda agar perubahan diterapkan.

AWS CLI

Perintah [update-integration](#) berikut memperbarui integrasi pribadi untuk menggunakan tautan VPC V2:

```
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\\"}]"
```

Integrasi pribadi menggunakan tautan VPC V1 (warisan)

Note

Implementasi integrasi pribadi berikut menggunakan tautan VPC V1. Tautan VPC V1 adalah sumber daya lama. Kami menyarankan Anda menggunakan [tautan VPC V2 untuk](#) REST APIs

Untuk membuat integrasi pribadi, Anda harus terlebih dahulu membuat Network Load Balancer. Network Load Balancer Anda harus memiliki [listener](#) yang merutekan permintaan ke sumber daya di VPC Anda. Untuk meningkatkan ketersediaan API Anda, pastikan Network Load Balancer merutekan lalu lintas ke sumber daya di lebih dari satu Availability Zone di Wilayah AWS. Kemudian, Anda membuat tautan VPC yang Anda gunakan untuk menghubungkan API dan Network Load Balancer Anda. Setelah membuat tautan VPC, Anda membuat integrasi pribadi untuk merutekan lalu lintas dari

API ke sumber daya di VPC melalui tautan VPC dan Network Load Balancer. Network Load Balancer dan API harus dimiliki oleh yang sama. Akun AWS

Topik

- [Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway \(lama\)](#)
- [Berikan izin untuk API Gateway untuk membuat tautan VPC \(lama\)](#)
- [Siapkan API Gateway API dengan integrasi pribadi menggunakan AWS CLI \(legacy\)](#)
- [Akun API Gateway yang digunakan untuk integrasi pribadi \(lama\)](#)

Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway (lama)

Note

Implementasi integrasi pribadi berikut menggunakan tautan VPC V1. Tautan VPC V1 adalah sumber daya lama. Kami menyarankan Anda menggunakan [tautan VPC V2 untuk REST APIs](#)

Prosedur berikut menguraikan langkah-langkah untuk menyiapkan Network Load Balancer (NLB) untuk integrasi pribadi API Gateway menggunakan konsol Amazon EC2 dan memberikan referensi untuk petunjuk terperinci untuk setiap langkah.

Untuk setiap VPC tempat Anda memiliki sumber daya, Anda hanya perlu mengkonfigurasi satu NLB dan satu VPCLink NLB mendukung beberapa [pendengar](#) dan grup [target](#) per NLB. Anda dapat mengonfigurasi setiap layanan sebagai pendengar tertentu di NLB dan menggunakan satu layanan VPCLink untuk terhubung ke NLB. Saat membuat integrasi pribadi di API Gateway, Anda kemudian menentukan setiap layanan menggunakan port spesifik yang ditetapkan untuk setiap layanan. Untuk informasi selengkapnya, lihat [the section called “Tutorial: Buat REST API dengan integrasi pribadi”](#). Network Load Balancer dan API harus dimiliki oleh yang sama. Akun AWS

Untuk membuat Network Load Balancer untuk integrasi pribadi menggunakan konsol API Gateway

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Siapkan server web pada instans Amazon EC2. Untuk contoh penyiapan, lihat [Menginstal Server Web LAMP di Amazon Linux 2](#).

3. Buat Network Load Balancer, daftarkan instans EC2 dengan grup target, dan tambahkan grup target ke pendengar Network Load Balancer. Untuk detailnya, ikuti petunjuk di [Memulai dengan Network Load Balancers](#).
4. Setelah Network Load Balancer dibuat, lakukan hal berikut:
 - a. Perhatikan ARN dari Network Load Balancer. Anda akan memerlukannya untuk membuat tautan VPC di API Gateway untuk mengintegrasikan API dengan sumber daya VPC di belakang Network Load Balancer.
 - b. Matikan evaluasi grup keamanan untuk PrivateLink.
 - Untuk menonaktifkan evaluasi grup keamanan untuk PrivateLink lalu lintas menggunakan konsol, Anda dapat memilih tab Keamanan, lalu Edit. Dalam pengaturan Keamanan, hapus Menegakkan aturan masuk pada PrivateLink lalu lintas.
 - Gunakan [set-security-groups](#) perintah berikut untuk mematikan evaluasi grup keamanan untuk PrivateLink lalu lintas:

```
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

Jangan menambahkan dependensi apa pun ke API Gateway CIDRs karena mereka pasti akan berubah tanpa pemberitahuan.

Berikan izin untuk API Gateway untuk membuat tautan VPC (lama)

Note

Implementasi integrasi pribadi berikut menggunakan tautan VPC V1. Tautan VPC V1 adalah sumber daya lama. Kami menyarankan Anda menggunakan [tautan VPC V2 untuk](#) REST APIs

Agar Anda atau pengguna di akun Anda dapat membuat dan memelihara tautan VPC, Anda atau pengguna harus memiliki izin untuk membuat, menghapus, dan melihat konfigurasi layanan titik akhir VPC, mengubah izin layanan titik akhir VPC, dan memeriksa penyeimbang beban. Untuk memberikan izin tersebut, gunakan langkah-langkah berikut.

Untuk memberikan izin untuk membuat, memperbarui, dan menghapus tautan VPC

1. Buat kebijakan IAM yang mirip dengan berikut ini:

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": "*"
  }
}

```

Jika Anda ingin mengaktifkan penandaan untuk tautan VPC Anda, pastikan untuk mengizinkan operasi penandaan. Untuk informasi selengkapnya, lihat [the section called “Izinkan operasi penandaan”](#).

2. Buat atau pilih peran IAM dan lampirkan kebijakan sebelumnya ke peran tersebut.
3. Tetapkan peran IAM kepada Anda atau pengguna di akun Anda yang membuat tautan VPC.

Siapkan API Gateway API dengan integrasi pribadi menggunakan AWS CLI (legacy)

Note

Implementasi integrasi pribadi berikut menggunakan tautan VPC V1. Tautan VPC V1 adalah sumber daya lama. Kami menyarankan Anda menggunakan [tautan VPC V2 untuk REST APIs](#).

Tutorial berikut menunjukkan cara menggunakan AWS CLI untuk membuat link VPC dan integrasi pribadi. Prasyarat berikut diperlukan:

- Anda memerlukan Network Load Balancer yang dibuat dan dikonfigurasi dengan sumber VPC Anda sebagai target. Untuk informasi selengkapnya, lihat [Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway \(lama\)](#). Ini harus Akun AWS sama dengan API Anda. Anda memerlukan Network Load Balancer ARN untuk membuat tautan VPC Anda.
- Untuk membuat dan mengelola VpcLink, Anda memerlukan izin untuk membuat VpcLink di API Anda. Anda tidak memerlukan izin untuk menggunakan file. VpcLink Untuk informasi selengkapnya, lihat [Berikan izin untuk API Gateway untuk membuat tautan VPC \(lama\)](#).

Untuk menyiapkan API dengan integrasi pribadi menggunakan AWS CLI

1. Gunakan [create-vpc-link](#) perintah berikut untuk membuat VpcLink penargetan Network Load Balancer yang ditentukan:

```
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
```

Output dari perintah ini mengakui penerimaan permintaan dan menunjukkan PENDING status untuk yang VpcLink sedang dibuat.

```
{  
  "status": "PENDING",  
  "targetArns": [  
    "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-  
vpcLink-test-nlb/1234567890abcdef"  
  ],  
  "id": "gim7c3",  
  "name": "my-test-vpc-link"  
}
```

Dibutuhkan 2-4 menit bagi API Gateway untuk menyelesaikan pembuatanVpcLink. Ketika operasi selesai dengan sukses, status adalahAVAILABLE. Anda dapat memverifikasi ini dengan menggunakan [get-vpc-link](#)perintah berikut:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Jika operasi gagal, Anda mendapatkan FAILED status, dengan statusMessage berisi pesan kesalahan. Misalnya, jika Anda mencoba membuat VpcLink dengan Network Load Balancer yang sudah dikaitkan dengan titik akhir VPC, Anda mendapatkan hal berikut di properti: statusMessage

```
"NLB is already associated with another VPC Endpoint Service"
```

Setelah berhasil VpcLink dibuat, Anda dapat membuat API dan mengintegrasikannya dengan sumber daya VPC melalui file. VpcLink

Perhatikan id nilai yang baru dibuatVpcLink. Dalam contoh output ini, itugim7c3. Anda membutuhkannya untuk mengatur integrasi pribadi.

- Gunakan [create-rest-api](#)perintah berikut untuk membuat [RestApi](#)resource API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Perhatikan nilai dan id `rootResourceId` nilai `RestApi` RestApi's dalam hasil yang dikembalikan. Anda memerlukan nilai ini untuk melakukan operasi lebih lanjut pada API.

Selanjutnya, Anda membuat API dengan hanya GET metode pada sumber daya root (/) dan mengintegrasikan metode dengan `VpcLink`.

- Gunakan perintah [put-method](#) berikut untuk membuat metode: GET /

```
aws apigateway put-method \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --authorization-type "NONE"
```

Jika Anda tidak menggunakan integrasi proxy dengan `VpcLink`, Anda juga harus menyiapkan setidaknya respons metode kode 200 status. Anda menggunakan integrasi proxy di sini.

- Setelah Anda membuat GET / metode, Anda mengatur integrasi. Untuk integrasi pribadi, Anda menggunakan `connection-id` parameter untuk memberikan `VpcLink` ID. Anda dapat menggunakan variabel tahap atau langsung memasukkan `VpcLink` ID. `uriParameter` ini tidak digunakan untuk merutekan permintaan ke titik akhir Anda, tetapi digunakan untuk mengatur Host header dan untuk validasi sertifikat.

Use the VPC link ID

Gunakan perintah [put-integration](#) berikut untuk menggunakan `VpcLink` ID secara langsung dalam integrasi:

```
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 gim7c3
```

Use a stage variable

Gunakan perintah [put-integration](#) berikut untuk menggunakan variabel stage untuk mereferensikan ID tautan VPC. Saat menerapkan API ke suatu panggung, Anda menyetel ID tautan VPC.

```
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}"
```

Pastikan untuk mengutip dua kali ekspresi variabel tahap (`\${stageVariables.vpcLinkId}`) dan melarikan diri dari `$` karakter.

Kapan pun, Anda juga dapat memperbarui integrasi untuk mengubah `connection-id`. Gunakan perintah [update-integration](#) berikut untuk memperbarui integrasi Anda:

```
aws apigateway update-integration \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --patch-operations '[{"op":"replace","path":"/  
connectionId","value":"\${stageVariables.vpcLinkId}"}]'
```

Pastikan untuk menggunakan daftar JSON stringified sebagai nilai parameter. `patch-operations`

Karena Anda menggunakan integrasi proxy pribadi, API Anda sekarang siap untuk penerapan dan untuk uji coba.

5. Jika Anda menggunakan variabel stage untuk menentukan `connection-id`, Anda perlu menerapkan API untuk mengujinya. Gunakan perintah [create-deployment](#) berikut untuk menerapkan API Anda dengan variabel stage:

```
aws apigateway create-deployment \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --variables vpcLinkId=gim7c3
```

Untuk memperbarui variabel stage dengan VpcLink ID yang berbeda, seperti *asf9d7*, gunakan perintah [update-stage](#) berikut:

```
aws apigateway update-stage \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Saat Anda membuat hardcode connection-id properti dengan VpcLink ID literal, Anda tidak perlu menerapkan API untuk mengujinya. Gunakan [test-invoke-method](#) perintah untuk menguji API Anda sebelum diterapkan.

- Gunakan perintah berikut untuk menjalankan API Anda:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

Atau, Anda dapat memasukkan URL Invoke-API Anda di browser web untuk melihat hasilnya.

Akun API Gateway yang digunakan untuk integrasi pribadi (lama)

Akun API Gateway khusus wilayah berikut secara otomatis IDs ditambahkan ke layanan titik akhir VPC Anda seperti AllowedPrincipals saat Anda membuat file. VpcLink

Wilayah	ID Akun
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407

Wilayah	ID Akun
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
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773

Wilayah	ID Akun
me-south-1	855739686837
me-central-1	614065512851

Integrasi tiruan untuk REST APIs di API Gateway

Amazon API Gateway mendukung integrasi tiruan untuk metode API. Fitur ini memungkinkan pengembang API untuk menghasilkan respons API dari API Gateway secara langsung, tanpa perlu backend integrasi. Sebagai pengembang API, Anda dapat menggunakan fitur ini untuk membuka blokir tim dependen yang perlu bekerja dengan API sebelum pengembangan proyek selesai. Anda juga dapat menggunakan fitur ini untuk menyediakan halaman landing untuk API Anda, yang dapat memberikan ikhtisar dan navigasi ke API Anda. Untuk contoh halaman arahan seperti itu, lihat permintaan integrasi dan respons metode GET pada sumber daya root dari contoh API yang dibahas [Tutorial: Buat REST API dengan mengimpor contoh](#).

Sebagai pengembang API, Anda memutuskan bagaimana API Gateway merespons permintaan integrasi tiruan. Untuk ini, Anda mengonfigurasi permintaan integrasi metode dan respons integrasi untuk mengaitkan respons dengan kode status tertentu. Untuk metode dengan integrasi tiruan untuk mengembalikan 200 respons, konfigurasi template pemetaan badan permintaan integrasi untuk mengembalikan yang berikut ini.

```
{"statusCode": 200}
```

Konfigurasi respons 200 integrasi untuk memiliki template pemetaan tubuh berikut, misalnya:

```
{
  "statusCode": 200,
  "message": "Go ahead without me."
}
```

Demikian pula, agar metode mengembalikan, misalnya, respons 500 kesalahan, siapkan templat pemetaan badan permintaan integrasi untuk mengembalikan yang berikut ini.

```
{"statusCode": 500}
```

Siapkan respons 500 integrasi dengan, misalnya, templat pemetaan berikut:

```
{
  "statusCode": 500,
  "message": "The invoked method is not supported on the API resource."
}
```

Atau, Anda dapat meminta metode integrasi tiruan mengembalikan respons integrasi default tanpa menentukan templat pemetaan permintaan integrasi. Respons integrasi default adalah respons dengan regex status HTTP yang tidak ditentukan. Pastikan perilaku passthrough yang sesuai ditetapkan.

Note

Integrasi tiruan tidak dimaksudkan untuk mendukung template respons besar. Jika Anda membutuhkannya untuk kasus penggunaan Anda, Anda harus mempertimbangkan untuk menggunakan integrasi Lambda sebagai gantinya.

Dengan menggunakan templat pemetaan permintaan integrasi, Anda dapat menyuntikkan logika aplikasi untuk memutuskan respons integrasi tiruan mana yang akan dikembalikan berdasarkan kondisi tertentu. Misalnya, Anda dapat menggunakan parameter scope kueri pada permintaan yang masuk untuk menentukan apakah akan mengembalikan respons yang berhasil atau respons kesalahan:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

Dengan cara ini, metode integrasi tiruan memungkinkan panggilan internal dilakukan sambil menolak jenis panggilan lain dengan respons kesalahan.

Di bagian ini, kami menjelaskan cara menggunakan konsol API Gateway untuk mengaktifkan integrasi tiruan untuk metode API.

Topik

- [Aktifkan integrasi tiruan menggunakan konsol API Gateway](#)

Aktifkan integrasi tiruan menggunakan konsol API Gateway

Anda harus memiliki metode yang tersedia di API Gateway. Ikuti petunjuk dalam [Tutorial: Membuat REST API dengan integrasi non-proxy HTTP](#).

1. Pilih sumber daya API dan pilih metode Buat.

Untuk membuat metode, lakukan hal berikut:

- a. Untuk jenis Metode, pilih metode.
- b. Untuk jenis Integrasi, pilih Mock.
- c. Pilih metode Buat.
- d. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
- e. Pilih parameter string kueri URL. Pilih Tambahkan string kueri dan untuk Nama, masukkan **scope**. Parameter kueri ini menentukan apakah pemanggil internal atau sebaliknya.
- f. Pilih Simpan.

2. Pada tab respons Metode, pilih Buat respons, lalu lakukan hal berikut:

- a. Untuk Status HTTP, masukkan **500**.
- b. Pilih Simpan.

3. Pada tab Permintaan integrasi, untuk pengaturan permintaan Integrasi, pilih Edit.

4. Pilih template Pemetaan, lalu lakukan hal berikut:

- a. Pilih Tambahkan templat pemetaan.
- b. Untuk jenis Konten, masukkan **application/json**.
- c. Untuk badan Template, masukkan yang berikut ini:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

- d. Pilih Simpan.
- 5. Pada tab Respons Integrasi, untuk Default - Respons pilih Edit.
- 6. Pilih template Pemetaan, lalu lakukan hal berikut:
 - a. Untuk jenis Konten, masukkan **application/json**.
 - b. Untuk badan Template, masukkan yang berikut ini:

```
{
  "statusCode": 200,
  "message": "Go ahead without me"
}
```

- c. Pilih Simpan.
- 7. Pilih Buat respons.

Untuk membuat respons 500, lakukan hal berikut:

- a. Untuk regex status HTTP, masukkan. **5\d{2}**
- b. Untuk status respons Metode, pilih **500**.
- c. Pilih Simpan.
- d. Untuk 5\d{2} - Respons, pilih Edit.
- e. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan.
- f. Untuk jenis Konten, masukkan **application/json**.
- g. Untuk badan Template, masukkan yang berikut ini:

```
{
  "statusCode": 500,
  "message": "The invoked method is not supported on the API resource."
}
```

- h. Pilih Simpan.
- 8. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab. Untuk menguji integrasi tiruan Anda, lakukan hal berikut:
 - a. Masukkan `scope=internal` di bawah String kueri. Pilih Uji. Hasil tes menunjukkan:

Request: `/?scope=internal`

Status: 200

Latency: 26 ms

Response Body

```
{
  "statusCode": 200,
  "message": "Go ahead without me"
}
```

Response Headers

```
{"Content-Type":"application/json"}
```

- b. Masukkan `scope=public` di bawah Query strings atau biarkan kosong. Pilih Uji. Hasil tes menunjukkan:

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"}
```

Anda juga dapat mengembalikan header dalam respons integrasi tiruan dengan terlebih dahulu menambahkan header ke respons metode dan kemudian menyiapkan pemetaan header dalam respons integrasi. Faktanya, beginilah cara konsol API Gateway mengaktifkan dukungan CORS dengan mengembalikan header yang diperlukan CORS.

Minta validasi untuk REST APIs di API Gateway

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi dasar permintaan API sebelum melanjutkan dengan permintaan integrasi. Ketika validasi gagal, API Gateway segera gagal permintaan, mengembalikan respons kesalahan 400 ke pemanggil, dan menerbitkan hasil validasi di Log. CloudWatch Ini mengurangi panggilan yang tidak perlu ke backend. Lebih penting lagi, ini memungkinkan Anda fokus pada upaya validasi khusus untuk aplikasi Anda. Anda dapat memvalidasi badan permintaan dengan memverifikasi bahwa parameter permintaan yang diperlukan valid dan non-null atau dengan menentukan skema model untuk validasi data yang lebih rumit.

Topik

- [Ikhtisar validasi permintaan dasar di API Gateway](#)
- [Model data untuk REST APIs](#)
- [Siapkan validasi permintaan dasar di API Gateway](#)
- [AWS CloudFormation template API sampel dengan validasi permintaan dasar](#)

Ikhtisar validasi permintaan dasar di API Gateway

API Gateway dapat melakukan validasi permintaan dasar, sehingga Anda dapat fokus pada validasi khusus aplikasi di backend. Untuk validasi, API Gateway memverifikasi salah satu atau kedua kondisi berikut:

- Parameter permintaan yang diperlukan dalam URI, string kueri, dan header permintaan masuk disertakan dan tidak kosong. API Gateway hanya memeriksa keberadaan parameter dan tidak memeriksa jenis atau formatnya.
- Payload permintaan yang berlaku mematuhi permintaan [skema JSON](#) yang dikonfigurasi dari metode untuk jenis konten tertentu. Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, setel tipe konten untuk model data Anda\$default.

Untuk mengaktifkan validasi, Anda menentukan aturan validasi dalam validator [permintaan](#), [menambahkan validator](#) ke [peta API validator permintaan, dan menetapkan validator](#) ke metode API individual.

Note

Minta validasi badan dan [Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway](#) merupakan dua topik terpisah. Jika payload permintaan tidak memiliki skema model yang cocok, Anda dapat memilih untuk melewati atau memblokir muatan asli. Untuk informasi selengkapnya, lihat [Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway](#).

Model data untuk REST APIs

Di API Gateway, model mendefinisikan struktur data payload. Di API Gateway, model didefinisikan menggunakan [draf skema JSON](#) 4. Objek JSON berikut adalah data sampel dalam contoh Pet Store.

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

Data berisi `id`, `type`, dan `price` hewan peliharaan. Model data ini memungkinkan Anda untuk:

- Gunakan validasi permintaan dasar.
- Buat template pemetaan untuk transformasi data.
- Buat tipe data yang ditentukan pengguna (UDT) saat Anda membuat SDK.


```

1 {
2   "$schema": "http://json-schema.org/draft-04/ ← 1
   schema#",
3   "title": "PetStoreModel", ← 2
4   "type": "object",
5   "required": [ "price", "type" ], ← 3
6   "properties": {
7     "id": {
8       "type": "integer" ← 4
9     },
10    "type": {
11      "type": "string",
12      "enum": [ "dog", "cat", "fish" ] ← 5
13    },
14    "price": { ← 6
15      "type": "number",
16      "minimum": 25.0,
17      "maximum": 500.0
18    }
19  }
20 }

```

Dalam model ini:

1. `$schema` Objek mewakili pengidentifikasi versi Skema JSON yang valid. Skema ini adalah draf Skema JSON v4.
2. `title` Objek adalah pengidentifikasi yang dapat dibaca manusia untuk model tersebut. Judul ini adalah `PetStoreModel`.
3. Kata kunci `required` validasi membutuhk `type`, dan `price` untuk validasi permintaan dasar.
4. Model `properties` tersebut adalah `id`, `type`, dan `price`. Setiap objek memiliki properti yang dijelaskan dalam model.
5. Objek hanya `type` dapat memiliki nilai `dog`, `cat`, atau `fish`.

6. Objek `price` adalah angka dan dibatasi dengan minimum 25 dan 500maximum.

PetStore model

```
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 }
```

Dalam model ini:

1. Pada baris 2, `$schema` objek mewakili pengenalan versi Skema JSON yang valid. Skema ini adalah draf Skema JSON v4.
2. Pada baris 3, `title` objek adalah pengidentifikasi yang dapat dibaca manusia untuk model tersebut. Judul ini adalah `PetStoreModel`.
3. Pada baris 5, kata kunci `required` validasi membutuhkan `type`, dan `price` untuk validasi permintaan dasar.
4. Pada baris 6 - 17, modelnya adalah `id`, `type`, dan `price`. `properties` Setiap objek memiliki properti yang dijelaskan dalam model.
5. Pada baris 12, objek hanya `type` dapat memiliki nilai `dog`, `cat`, atau `fish`.
6. Pada baris 14 - 17, objek `price` adalah angka dan dibatasi dengan minimum 25 dan 500maximum.

Membuat model yang lebih kompleks

Anda dapat menggunakan `$ref` primitif untuk membuat definisi yang dapat digunakan kembali untuk model yang lebih panjang. Misalnya, Anda dapat membuat definisi yang disebut `Price` di `definitions` bagian yang menjelaskan `price` objek. Nilai `$ref` adalah `Price` definisi.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
      "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
    "Price": {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Anda juga dapat mereferensikan skema model lain yang ditentukan dalam file model eksternal. Tetapkan nilai `$ref` properti ke lokasi model. Dalam contoh berikut, `Price` model didefinisikan dalam `PetStorePrice` model di `APIa1234`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

```
}
```

Model yang lebih panjang dapat merujuk PetStorePrice model.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
      "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Menggunakan model data keluaran

Jika Anda mengubah data, Anda dapat menentukan model payload dalam respons integrasi. Model payload dapat digunakan saat Anda membuat SDK. Untuk bahasa yang diketik dengan kuat, seperti Java, Objective-C, atau Swift, objek sesuai dengan tipe data yang ditentukan pengguna (UDT). API Gateway membuat UDT jika Anda menyediakannya dengan model data saat Anda membuat SDK. Untuk informasi selengkapnya tentang transformasi data, lihat [Memetakan transformasi template untuk REST API di API Gateway](#).

Contoh berikut adalah data keluaran dari respons integrasi.

```
{
  [
    {
      "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
  }
]
```

Contoh berikut adalah model payload yang menggambarkan data output.

```
{
  "$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
    }
  }
}
```

Dengan model ini, Anda dapat memanggil SDK untuk mengambil nilai `description` dan `askingPrice` properti dengan membaca properti `PetStoreOutputModel[i].description` dan `PetStoreOutputModel[i].askingPrice`. Jika tidak ada model yang disediakan, API Gateway menggunakan model kosong untuk membuat UDT default.

Langkah selanjutnya

- Bagian ini menyediakan sumber daya yang dapat Anda gunakan untuk mendapatkan lebih banyak pengetahuan tentang konsep yang disajikan dalam topik ini.

Anda dapat mengikuti tutorial validasi permintaan:

- [Siapkan validasi permintaan menggunakan konsol API Gateway](#)
- [Siapkan validasi permintaan dasar menggunakan AWS CLI](#)

- [Siapkan validasi permintaan dasar menggunakan definisi OpenAPI](#)
- Untuk informasi selengkapnya tentang templat transformasi dan pemetaan data, [Memetakan transformasi template untuk REST API di API Gateway](#)

Siapkan validasi permintaan dasar di API Gateway

Bagian ini menunjukkan cara menyiapkan validasi permintaan untuk API Gateway menggunakan konsol AWS CLI, dan definisi OpenAPI.

Topik

- [Siapkan validasi permintaan menggunakan konsol API Gateway](#)
- [Siapkan validasi permintaan dasar menggunakan AWS CLI](#)
- [Siapkan validasi permintaan dasar menggunakan definisi OpenAPI](#)

Siapkan validasi permintaan menggunakan konsol API Gateway

Anda dapat menggunakan konsol API Gateway untuk memvalidasi permintaan dengan memilih salah satu dari tiga validator untuk permintaan API:

- Validasi tubuh.
- Validasi parameter string kueri dan header.
- Validasi isi, parameter string kueri, dan header.

Saat Anda menerapkan salah satu validator pada metode API, konsol API Gateway menambahkan validator ke peta API. [RequestValidators](#)

Untuk mengikuti tutorial ini, Anda akan menggunakan CloudFormation template untuk membuat API Gateway API yang tidak lengkap. API ini memiliki `/validator` sumber daya dengan GET dan POST metode. Kedua metode terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan mengkonfigurasi dua jenis validasi permintaan:

- Dalam GET metode ini, Anda akan mengkonfigurasi validasi permintaan untuk parameter string kueri URL.
- Dalam POST metode ini, Anda akan mengonfigurasi validasi permintaan untuk badan permintaan.

Ini akan memungkinkan hanya panggilan API tertentu untuk melewati API.

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#). Anda akan menggunakan template ini untuk membuat API yang tidak lengkap. Anda akan menyelesaikan langkah-langkah lainnya di konsol API Gateway.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **request-validation-tutorial-console** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Untuk memilih API yang baru dibuat

1. Pilih **request-validation-tutorial-console** tumpukan yang baru dibuat.
2. Pilih Sumber daya.
3. Di bawah Physical ID, pilih API Anda. Tautan ini akan mengarahkan Anda ke konsol API Gateway.

Sebelum Anda memodifikasi GET dan POST metode, Anda harus membuat model.

Untuk membuat model

1. Model diperlukan untuk menggunakan validasi permintaan pada badan permintaan yang masuk. Untuk membuat model, di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **PetStoreModel**.
4. Untuk Jenis Konten, masukkan **application/json**. Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **\$default**.
5. Untuk Deskripsi, masukkan **My PetStore Model** sebagai deskripsi model.
6. Untuk skema Model, tempelkan model berikut ke editor kode, dan pilih Buat.

```
{
  "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
    }
  }
}
```

Untuk informasi lebih lanjut tentang model, lihat [Model data untuk REST APIs](#).

Untuk mengkonfigurasi validasi permintaan untuk metode GET

1. Di panel navigasi utama, pilih Resources, lalu pilih metode GET.

2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header.
4. Di bawah parameter string kueri URL, lakukan hal berikut:
 - a. Pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan **petType**.
 - c. Aktifkan Diperlukan.
 - d. Tetap caching dimatikan.
5. Pilih Simpan.
6. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
7. Di bawah parameter string kueri URL, lakukan hal berikut:
 - a. Pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan **petType**.
 - c. Untuk Dipetakan dari, masukkan **method.request.querystring.petType**. Ini memetakan **petType** ke jenis hewan peliharaan.

Untuk informasi selengkapnya tentang pemetaan data, lihat tutorial [pemetaan data](#).
 - d. Tetap caching dimatikan.
8. Pilih Simpan.

Untuk menguji validasi permintaan untuk metode ini **GET**

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Kueri, masukkan **petType=dog**, lalu pilih Uji.
3. Tes metode akan kembali 200 OK dan daftar anjing-anjingnya.

Untuk informasi tentang cara mengubah data keluaran ini, lihat [tutorial pemetaan data](#).

4. Hapus **petType=dog** dan pilih Uji.
5. Tes metode akan mengembalikan 400 kesalahan dengan pesan kesalahan berikut:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Untuk mengkonfigurasi validasi permintaan untuk metode **POST**

1. Di panel navigasi utama, pilih Resources, lalu pilih metode POST.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi isi.
4. Di bawah badan Permintaan, pilih Tambah model.
5. Untuk jenis Konten, masukkan **application/json**. Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan `$default`.

Untuk Model, pilih PetStoreModel.

6. Pilih Simpan.

Untuk menguji validasi permintaan untuk metode **POST**

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk Request body paste berikut ini ke editor kode:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Pilih Uji.

3. Tes metode akan kembali 200 OK dan pesan sukses.
4. Untuk Request body paste berikut ini ke editor kode:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Pilih Uji.

5. Tes metode akan mengembalikan 400 kesalahan dengan pesan kesalahan berikut:

```
{
  "message": "Invalid request body"
}
```

Di bagian bawah log pengujian, alasan untuk badan permintaan yang tidak valid dikembalikan. Dalam hal ini, harga hewan peliharaan berada di luar maksimum yang ditentukan dalam model.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Langkah selanjutnya

- Untuk informasi tentang cara mengubah data keluaran dan melakukan lebih banyak pemetaan data, lihat tutorial [pemetaan data](#).
- Ikuti [Mengatur validasi permintaan dasar menggunakan AWS CLI](#) tutorial, untuk melakukan langkah-langkah serupa menggunakan. AWS CLI

Siapkan validasi permintaan dasar menggunakan AWS CLI

Anda dapat membuat validator untuk menyiapkan validasi permintaan menggunakan. AWS CLI Untuk mengikuti tutorial ini, Anda akan menggunakan CloudFormation template untuk membuat API Gateway API yang tidak lengkap.

Note

Ini bukan CloudFormation template yang sama dengan tutorial konsol.

Menggunakan `/validator` sumber daya pra-ekspos, Anda akan membuat GET dan POST metode. Kedua metode akan terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan mengkonfigurasi dua validasi permintaan berikut:

- Pada GET metode ini, Anda akan membuat `params-only` validator untuk memvalidasi parameter string kueri URL.
- Pada POST metode ini, Anda akan membuat `body-only` validator untuk memvalidasi badan permintaan.

Ini akan memungkinkan hanya panggilan API tertentu untuk melewati API.

Untuk membuat CloudFormation tumpukan

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#).

Untuk menyelesaikan tutorial berikut, Anda memerlukan [AWS Command Line Interface \(AWS CLI\) versi 2](#).

Untuk perintah panjang, karakter escape (`\`) digunakan untuk memisahkan perintah menjadi beberapa baris.

Note

Di Windows, beberapa perintah Bash CLI yang biasa Anda gunakan (zipseperti) tidak didukung oleh terminal bawaan sistem operasi. Untuk mendapatkan versi terintegrasi Windows dari Ubuntu dan Bash, [instal Windows Subsystem untuk Linux](#). Contoh perintah CLI dalam panduan ini menggunakan pemformatan Linux. Perintah yang menyertakan dokumen JSON sebaris harus diformat ulang jika Anda menggunakan CLI Windows.

1. Gunakan perintah berikut untuk membuat CloudFormation tumpukan.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
--template-body file://request-validation-tutorial-cli.zip --capabilities
CAPABILITY_NAMED_IAM
```

2. CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Gunakan perintah berikut untuk melihat status CloudFormation tumpukan Anda.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Ketika status CloudFormation tumpukan Anda `StackStatus: "CREATE_COMPLETE"`, gunakan perintah berikut untuk mengambil nilai output yang relevan untuk langkah-langkah masa depan.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli  
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,  
Description: Description}"
```

Nilai output adalah sebagai berikut:

- Apild, yang merupakan ID untuk API. Untuk tutorial ini, ID API adalah abc123.
- ResourceId, yang merupakan ID untuk sumber daya validator tempat GET dan POST metode diekspos. Untuk tutorial ini, ID Sumber Daya adalah efg456

Untuk membuat validator permintaan dan mengimpor model

1. Validator diperlukan untuk menggunakan validasi permintaan dengan file. AWS CLI Gunakan perintah berikut untuk membuat validator yang hanya memvalidasi parameter permintaan.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
--no-validate-request-body \  
--validate-request-parameters \  
--name params-only
```

Perhatikan ID params-only validator.

2. Gunakan perintah berikut untuk membuat validator yang hanya memvalidasi badan permintaan.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
--validate-request-body \  
--no-validate-request-parameters \  
--name body-only
```

Perhatikan ID body-only validator.

3. Model diperlukan untuk menggunakan validasi permintaan pada badan permintaan yang masuk. Gunakan perintah berikut untuk mengimpor model.

```
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} } }
```

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Untuk membuat **GET** dan **POST** metode

1. Gunakan perintah berikut untuk menambahkan metode GET HTTP pada `/validate` sumber daya. Perintah ini menciptakan GET metode, menambahkan `params-only` validator, dan menetapkan string query `petType` sesuai kebutuhan.

```
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"
```

Gunakan perintah berikut untuk menambahkan metode POST HTTP pada `/validate` sumber daya. Perintah ini membuat POST metode, menambahkan `body-only` validator, dan melampirkan model ke validator `body-only`.

```
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. Gunakan perintah berikut untuk mengatur 200 OK respons GET `/validate` metode.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
  --status-code 200
```

Gunakan perintah berikut untuk mengatur 200 OK respons POST `/validate` metode.

```
aws apigateway put-method-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --status-code 200
```

3. Gunakan perintah berikut untuk mengatur Integration dengan titik akhir HTTP tertentu untuk GET /validation metode ini.

```
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'
```

Gunakan perintah berikut untuk mengatur Integration dengan titik akhir HTTP tertentu untuk POST /validation metode ini.

```
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. Gunakan perintah berikut untuk menyiapkan respons integrasi untuk GET /validation metode ini.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --status-code 200 \  
    --selection-pattern ""
```

Gunakan perintah berikut untuk menyiapkan respons integrasi untuk POST /validation metode ini.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
    --resource-id efg456
```

```
--resource-id efg456 \  
--http-method POST \  
--status-code 200 \  
--selection-pattern ""
```

Untuk menguji API

1. Untuk menguji GET metode, yang akan melakukan validasi permintaan untuk string query, gunakan perintah berikut:

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method GET \  
--path-with-query-string '/validate?petType=dog'
```

Hasilnya akan mengembalikan daftar 200 OK dan anjing-anjingnya.

2. Gunakan perintah berikut untuk menguji tanpa menyertakan string kueri petType

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method GET
```

Hasilnya akan mengembalikan 400 kesalahan.

3. Untuk menguji POST metode, yang akan melakukan validasi permintaan untuk badan permintaan, gunakan perintah berikut:

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method POST \  
--body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

Hasilnya akan mengembalikan pesan 200 OK dan sukses.

4. Gunakan perintah berikut untuk menguji menggunakan badan yang tidak valid.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method POST \  
--body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```


Hasilnya akan mengembalikan 400 kesalahan, karena harga anjingnya melebihi harga maksimum yang ditentukan oleh model.

Untuk menghapus CloudFormation tumpukan

- Gunakan perintah berikut untuk menghapus CloudFormation sumber daya Anda.

```
aws cloudformation delete-stack --stack-name request-validation-tutorial-cli
```

Siapkan validasi permintaan dasar menggunakan definisi OpenAPI

Anda dapat mendeklarasikan validator permintaan di API level dengan menentukan satu set [x-amazon-apigateway-request-Validators.requestValidator objek](#) objek di [x-amazon-apigateway-request-validator objek](#) peta untuk memilih bagian permintaan mana yang akan divalidasi. Dalam definisi OpenAPI contoh, ada dua validator:

- `allvalidator` yang memvalidasi tubuh, menggunakan model `RequestBodyModel` data, dan parameter.

Model `RequestBodyModel` data mengharuskan objek input JSON berisiname, type, dan price properti. `nameProperti` dapat berupa string apa pun, `type` harus menjadi salah satu bidang enumerasi yang ditentukan (`["dog", "cat", "fish"]`), dan `price` harus berkisar antara 25 dan 500. `idParameter` tidak diperlukan.

- `param-only` yang hanya memvalidasi parameter.

Untuk mengaktifkan validator permintaan pada semua metode API, tentukan [x-amazon-apigateway-requestproperty -validator](#) properti di tingkat API definisi OpenAPI. Dalam definisi OpenAPI contoh, `all` validator digunakan pada semua metode API, kecuali jika diganti. Saat menggunakan model untuk memvalidasi isi, jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Untuk mengaktifkan validator permintaan pada metode individual, tentukan `x-amazon-apigateway-request-validator` properti di tingkat metode. Dalam contoh, definisi OpenAPI, `param-only` validator menimpa validator pada metode. `all` GET

Untuk mengimpor contoh OpenAPI ke API Gateway, lihat petunjuk berikut ke [Impor API Regional ke API Gateway](#) atau ke [Impor API yang dioptimalkan tepi ke API Gateway](#)

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 template API sampel dengan validasi permintaan dasar

CloudFormation Contoh definisi template berikut mendefinisikan API sampel dengan validasi permintaan diaktifkan. API adalah bagian dari [PetStoreAPI](#). Ini mengekspos POST metode untuk menambahkan hewan peliharaan ke pets koleksi dan GET metode untuk menanyakan hewan peliharaan dengan jenis tertentu.

Ada dua validator permintaan yang dideklarasikan:

GETValidator

Validator ini diaktifkan pada metode. GET Ini memungkinkan API Gateway untuk memverifikasi bahwa parameter kueri yang diperlukan (q1) disertakan dan tidak kosong dalam permintaan yang masuk.

POSTValidator

Validator ini diaktifkan pada metode. POST Ini memungkinkan API Gateway untuk memverifikasi bahwa format permintaan payload mematuhi yang ditentukan `RequestBodyModel` ketika jenis konten `application/json` jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default`. `RequestBodyModel` berisi model tambahan, `RequestBodyModelId`, untuk menentukan ID hewan peliharaan.

```
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}'

```

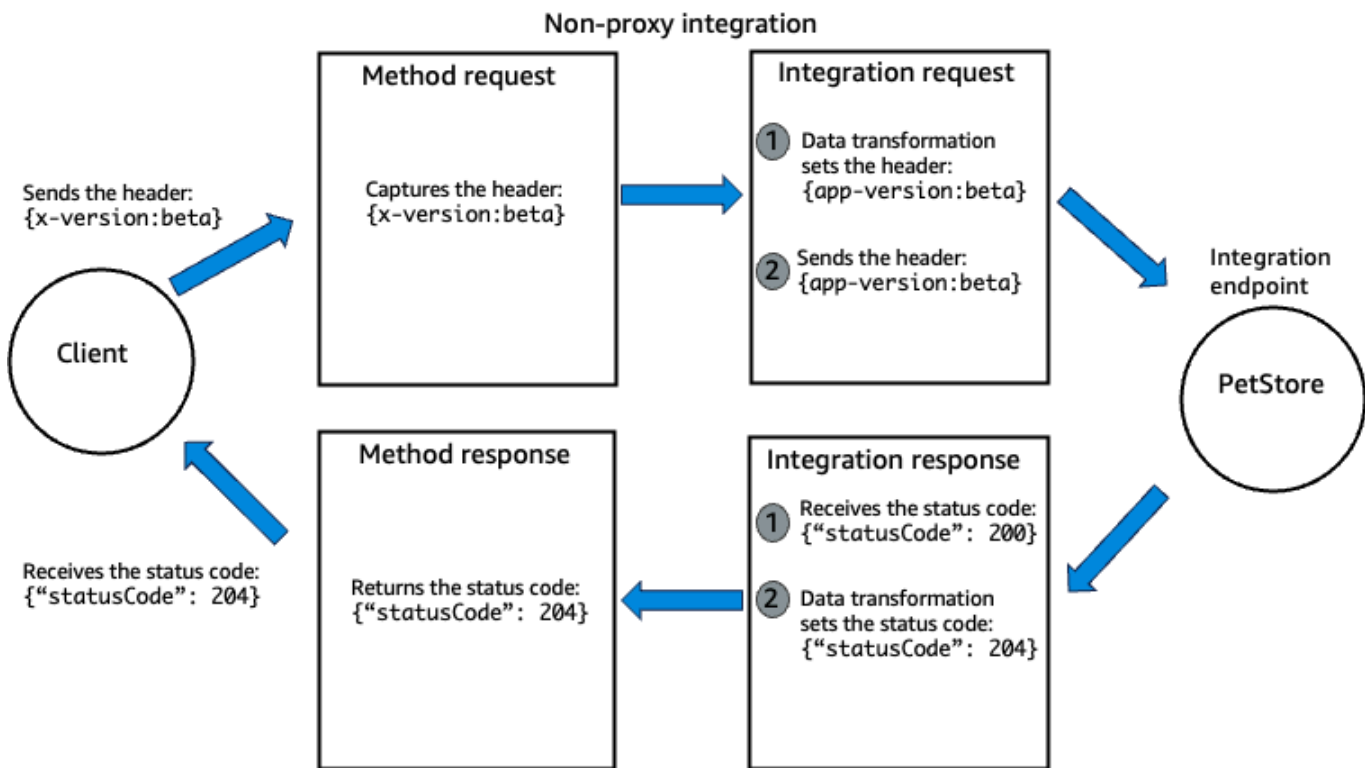
Transformasi data untuk REST API di API Gateway

Note

Bagian ini menjelaskan fitur yang Anda gunakan dengan integrasi non-proxy. Namun, kami menyarankan jika memungkinkan, Anda menggunakan integrasi proxy untuk REST API Anda. Integrasi proxy memiliki pengaturan integrasi yang efisien dan dapat berkembang dengan backend tanpa harus meruntuhkan pengaturan yang ada. Untuk informasi selengkapnya, lihat [the section called “Pilih jenis integrasi API”](#).

Jika Anda menggunakan integrasi non-proxy, Anda dapat menggunakan dua fitur API Gateway untuk mengubah permintaan metode dan respons integrasi Anda. Anda dapat mengubah permintaan metode jika membutuhkan format payload yang berbeda dari payload permintaan integrasi. Anda dapat mengubah respons integrasi Anda jika mengembalikan format payload yang berbeda dari format yang perlu Anda kembalikan dalam respons metode. Untuk informasi selengkapnya tentang siklus hidup permintaan, lihat [the section called “Contoh sumber daya untuk REST API”](#)

Contoh berikut menunjukkan transformasi data di mana untuk header "x-version:beta", parameter x-version header diubah menjadi parameter app-version header. Transformasi data dari x-version ke app-version terjadi dalam permintaan integrasi. Dengan begitu, titik akhir integrasi menerima nilai parameter header yang diubah. Ketika titik akhir integrasi mengembalikan kode status, kode status diubah dari 200 ke 204 sebelum respons metode.



Untuk membuat transformasi data, Anda dapat menggunakan fitur berikut:

Pemetaan parameter

Dalam pemetaan parameter, Anda dapat mengubah parameter jalur URL permintaan integrasi, parameter string kueri URL, atau nilai header HTTP, tetapi Anda tidak dapat mengubah payload permintaan integrasi. Anda juga dapat memodifikasi nilai header respons HTTP. Gunakan pemetaan parameter untuk membuat nilai header statis untuk berbagi sumber daya lintas asal (CORS).

Anda dapat menggunakan pemetaan parameter dalam permintaan integrasi untuk integrasi proxy dan non-proxy, tetapi untuk menggunakan pemetaan parameter untuk respons integrasi, Anda memerlukan integrasi non-proxy. Pemetaan parameter tidak memerlukan skrip apa pun di [Velocity Template Language \(VTL\)](#). Untuk informasi selengkapnya, lihat [the section called “Pemetaan parameter”](#).

Pemetaan transformasi template

Dalam transformasi template pemetaan, Anda menggunakan template pemetaan untuk memetakan parameter jalur URL, parameter string kueri URL, header HTTP, dan permintaan integrasi atau badan respons integrasi. Template pemetaan adalah skrip yang dinyatakan dalam

[Velocity Template Language \(VTL\)](#) menggunakan [ekspresi JsonPath](#) dan diterapkan ke payload berdasarkan header. Content-type

Dengan template pemetaan, Anda dapat melakukan hal berikut:

- Pilih data mana yang akan dikirim menggunakan integrasi Layanan AWS, seperti fungsi Amazon DynamoDB atau Lambda, atau titik akhir HTTP. Untuk informasi selengkapnya, lihat [the section called “Tutorial: Memodifikasi permintaan integrasi dan respon untuk integrasi AWS layanan”](#).
- Ganti permintaan integrasi API dan parameter respons integrasi secara kondisional, buat nilai header baru, dan ganti kode status. Untuk informasi selengkapnya, lihat [the section called “Ganti parameter permintaan dan respons API Anda serta kode status”](#).

Anda juga dapat menentukan perilaku API Anda ketika badan permintaan integrasi memiliki Content-type header tanpa templat pemetaan yang cocok. Ini disebut perilaku passthrough integrasi. Untuk informasi selengkapnya, lihat [the section called “Perilaku permintaan metode untuk muatan tanpa templat pemetaan”](#).

Pilih antara pemetaan parameter dan transformasi template pemetaan

Kami menyarankan Anda menggunakan pemetaan parameter untuk mengubah data Anda bila memungkinkan. Jika API Anda mengharuskan Anda mengubah isi, atau mengharuskan Anda melakukan penggantian dan modifikasi bersyarat berdasarkan permintaan integrasi atau respons integrasi yang masuk, dan Anda tidak dapat menggunakan integrasi proxy, gunakan transformasi templat pemetaan.

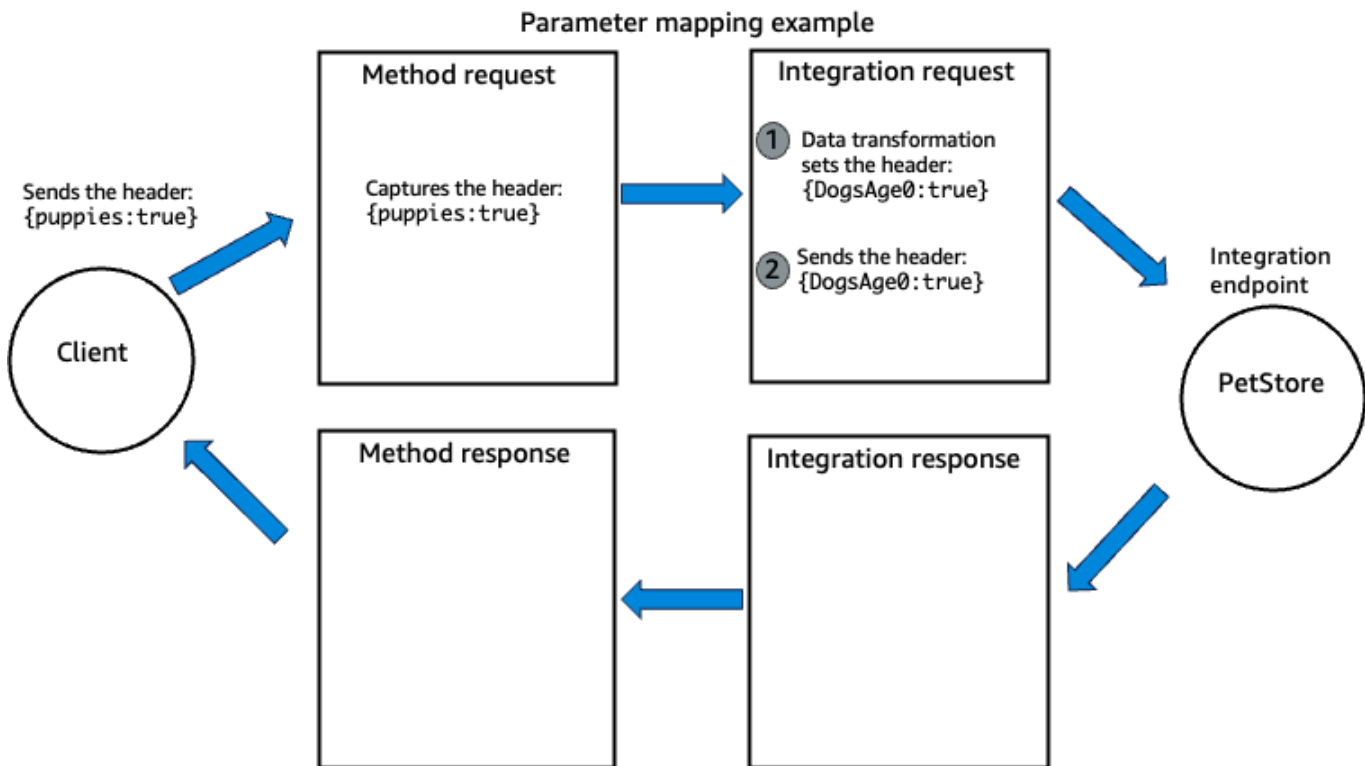
Pemetaan parameter untuk REST API di API Gateway

Note

Jika Anda menggunakan API HTTP, lihat [the section called “Pemetaan parameter”](#).

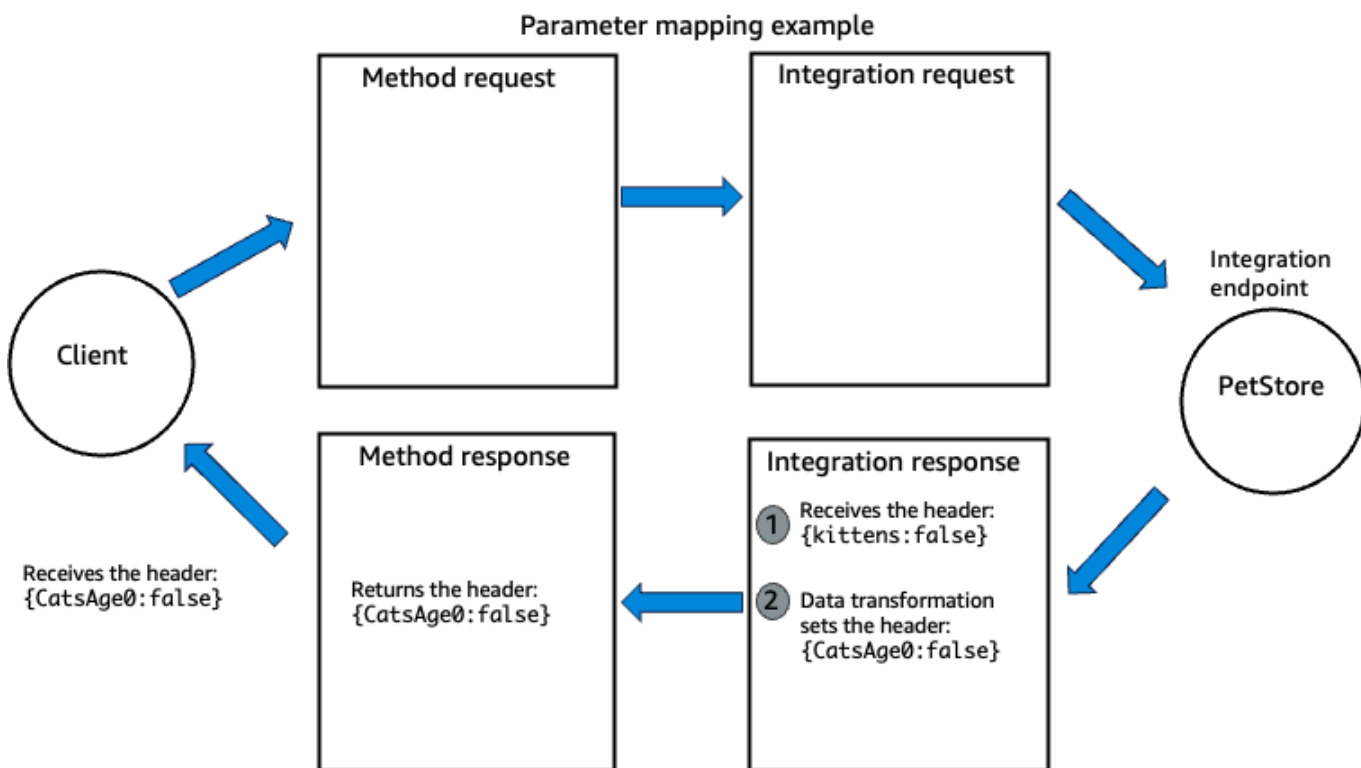
Dalam pemetaan parameter, Anda memetakan parameter permintaan atau respons. Anda dapat memetakan parameter menggunakan ekspresi pemetaan parameter atau nilai statis. Untuk daftar ekspresi pemetaan, lihat [the section called “Referensi sumber pemetaan parameter”](#). Anda dapat menggunakan pemetaan parameter dalam permintaan integrasi untuk integrasi proxy dan non-proxy, tetapi untuk menggunakan pemetaan parameter untuk respons integrasi, Anda memerlukan integrasi non-proxy.

Misalnya, Anda dapat memetakan parameter header permintaan metode puppies ke parameter header permintaan integrasiDogsAge0. Kemudian, jika klien mengirim header puppies:true ke API Anda, permintaan integrasi mengirimkan header permintaan DogsAge0:true ke titik akhir integrasi. Diagram berikut menunjukkan siklus hidup permintaan dari contoh ini.



Untuk membuat contoh ini menggunakan API Gateway, lihat [the section called “Contoh 1: Petakan parameter permintaan metode ke parameter permintaan integrasi”](#).

Sebagai contoh lain, Anda juga dapat memetakan parameter header respons integrasi kittens ke parameter header respons metodeCatsAge0. Kemudian, jika titik akhir integrasi kembalikittens:false, klien menerima headerCatsAge0:false. Diagram berikut menunjukkan siklus hidup permintaan dari contoh ini.



Topik

- [Contoh pemetaan parameter untuk REST API di API Gateway](#)
- [Referensi sumber pemetaan parameter untuk REST API di API Gateway](#)

Contoh pemetaan parameter untuk REST API di API Gateway

Contoh berikut menunjukkan cara membuat ekspresi pemetaan parameter menggunakan konsol API Gateway, OpenAPI, CloudFormation dan template. Untuk contoh cara menggunakan pemetaan parameter untuk membuat header CORS yang diperlukan, lihat. [the section called “CORS”](#)

Contoh 1: Petakan parameter permintaan metode ke parameter permintaan integrasi

Contoh berikut memetakan parameter header permintaan metode puppies ke parameter header permintaan integrasi DogsAge0.

Konsol Manajemen AWS

Untuk memetakan parameter permintaan metode

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pilih REST API.
3. Pilih metode.

Metode Anda harus memiliki integrasi non-proxy.

4. Untuk pengaturan permintaan Metode, pilih Edit.
5. Pilih header permintaan HTTP.
6. Pilih Tambahkan header.
7. Untuk Nama, masukkan **puppies**.
8. Pilih Simpan.
9. Pilih tab Permintaan integrasi, dan kemudian untuk pengaturan permintaan Integrasi, pilih Edit.

Konsol Manajemen AWS Secara otomatis menambahkan pemetaan parameter dari `method.request.header.puppies` ke `puppies` untuk Anda, tetapi Anda perlu mengubah Nama agar sesuai dengan parameter header permintaan yang diharapkan oleh titik akhir integrasi Anda.

10. Untuk Nama, masukkan **DogsAge0**.
11. Pilih Simpan.
12. Menerapkan ulang API Anda agar perubahan diterapkan.

Langkah-langkah berikut menunjukkan cara memverifikasi bahwa pemetaan parameter Anda berhasil.

(Opsional) Uji pemetaan parameter Anda

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk header, masukkan **puppies:true**.
3. Pilih Uji.
4. Dalam Log, hasilnya akan terlihat seperti berikut:

```
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}
```

Parameter header permintaan telah berubah dari `puppies` menjadi `DogsAge0`.

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

```
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"
  }
}
```

Contoh 2: Petakan beberapa parameter permintaan metode ke parameter permintaan integrasi yang berbeda

Contoh berikut memetakan parameter string permintaan metode multi-nilai `methodRequestQueryParam` ke parameter string kueri permintaan integrasi `integrationQueryParam` dan memetakan parameter header permintaan metode `methodRequestHeaderParam` ke parameter `integrationPathParam` jalur permintaan integrasi.

Konsol Manajemen AWS

Untuk memetakan parameter permintaan metode

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih metode.

Metode Anda harus memiliki integrasi non-proxy.

4. Untuk pengaturan permintaan Metode, pilih Edit.
5. Pilih parameter string kueri URL.
6. Pilih Tambahkan string kueri.
7. Untuk Nama, masukkan **methodRequestQueryParam**.
8. Pilih header permintaan HTTP.

9. Pilih Tambahkan header.
10. Untuk Nama, masukkan **methodRequestHeaderParam**.
11. Pilih Simpan.
12. Pilih tab Permintaan integrasi, dan kemudian untuk pengaturan permintaan Integrasi, pilih Edit.
13. Pilih parameter jalur URL.
14. Pilih Tambahkan parameter jalur.
15. Untuk Nama, masukkan **integrationPathParam**.
16. Untuk Dipetakan dari, masukkan **method.request.header.methodRequestHeaderParam**.

Ini memetakan header permintaan metode yang Anda tentukan dalam permintaan metode ke parameter jalur permintaan integrasi baru.

17. Pilih parameter string kueri URL.
18. Pilih Tambahkan string kueri.
19. Untuk Nama, masukkan **integrationQueryParam**.
20. Untuk Dipetakan dari, masukkan **method.request.multivaluequerystring.methodRequestQueryParam**.

Ini memetakan parameter string kueri multivalue ke parameter string kueri permintaan integrasi bernilai tunggal baru.

21. Pilih Simpan.
22. Menerapkan ulang API Anda agar perubahan diterapkan.

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

Definisi OpenAPI berikut membuat pemetaan parameter berikut untuk integrasi HTTP:

- Header permintaan metode, bernama **methodRequestHeaderParam**, ke dalam parameter jalur permintaan integrasi, bernama **integrationPathParam**
- Metode multi-nilai meminta string kueri, bernama **methodRequestQueryParam**, ke dalam string permintaan permintaan integrasi, bernama **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

Definisi OpenAPI berikut membuat pemetaan parameter berikut untuk integrasi HTTP:

- Header permintaan metode, bernamamethodRequestHeaderParam, ke dalam parameter jalur permintaan integrasi, bernama integrationPathParam
- Metode multi-nilai meminta string kueri, bernamamethodRequestQueryParam, ke dalam string permintaan permintaan integrasi, bernama 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"
    }
}
}
}
}

```

Contoh 3: Memetakan bidang dari badan permintaan JSON ke parameter permintaan integrasi

Anda juga dapat memetakan parameter permintaan integrasi dari bidang di badan permintaan JSON menggunakan ekspresi [JSONPath](#). Contoh berikut memetakan badan permintaan metode ke header permintaan integrasi bernama `body-header` dan memetakan bagian dari badan permintaan, seperti yang dinyatakan oleh ekspresi JSON ke header permintaan integrasi bernama `pet-price`.

Untuk menguji contoh ini, berikan masukan yang berisi kategori harga, seperti berikut ini:

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

Konsol Manajemen AWS

Untuk memetakan parameter permintaan metode

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih POST, PUT/PATCH, atau ANY metode.

Metode Anda harus memiliki integrasi non-proxy.

4. Untuk pengaturan permintaan Integrasi, pilih Edit.
5. Pilih parameter header permintaan URL.
6. Pilih Tambahkan parameter header permintaan.
7. Untuk Nama, masukkan **body-header**.
8. Untuk Dipetakan dari, masukkan **method.request.body**.

Ini memetakan badan permintaan metode ke parameter header permintaan integrasi baru.

9. Pilih Tambahkan parameter header permintaan.
10. Untuk Nama, masukkan **pet-price**.
11. Untuk Dipetakan dari, masukkan **method.request.body[0].price**.

Ini memetakan bagian dari badan permintaan metode ke parameter header permintaan integrasi baru.

12. Pilih Simpan.
13. Menerapkan ulang API Anda agar perubahan diterapkan.

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

```
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

Parameter permintaan integrasi peta definisi OpenAPI berikut dari bidang di badan permintaan JSON.

```

{
  "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"  
      }  
    }  
  }  
}
```

Contoh 4: Petakan respons integrasi terhadap respons metode

Anda juga dapat memetakan respons integrasi ke respons metode. Contoh berikut memetakan badan respons integrasi ke header respons metode bernama `location`, memetakan header respons integrasi `x-app-id` ke header respons metode `id`, dan memetakan header respons integrasi multi-nilai `item` ke header `items` respons metode.

Konsol Manajemen AWS

Untuk memetakan respon integrasi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pilih metode.

Metode Anda harus memiliki integrasi non-proxy.

4. Pilih tab respons Metode, dan kemudian untuk Respons 200, pilih Edit.
5. Untuk nama Header, pilih Tambahkan header.
6. Buat tiga header bernama `id`, `item`, dan `location`.
7. Pilih Simpan.
8. Pilih tab Integrasi respon, dan kemudian untuk Default - Respons, pilih Edit.
9. Di bawah pemetaan Header, masukkan yang berikut ini.
 - a. Untuk id, masukkan `integration.response.header.x-app-id`
 - b. Untuk item, masukkan `integration.response.multivalueheader.item`
 - c. Untuk lokasi, masukkan `integration.response.body.redirect.url`
10. Pilih Simpan.
11. Menerapkan ulang API Anda agar perubahan diterapkan.

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

```
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

Definisi OpenAPI berikut memetakan respons integrasi terhadap respons metode.

```

{
  "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
}
}
}
}

```


Referensi sumber pemetaan parameter untuk REST API di API Gateway

Saat Anda membuat pemetaan parameter, Anda menentukan permintaan metode atau parameter respons integrasi untuk dimodifikasi dan Anda menentukan cara memodifikasi parameter tersebut.

Tabel berikut menunjukkan parameter permintaan metode yang dapat Anda petakan, dan ekspresi untuk membuat pemetaan. Dalam ekspresi ini, *name* adalah nama parameter permintaan metode. Misalnya, untuk memetakan parameter header permintaan `puppies`, gunakan ekspresi `method.request.header.puppies`. Ekspresi Anda harus sesuai dengan ekspresi reguler `^[a-zA-Z0-9._$-]+$`. Anda dapat menggunakan pemetaan parameter dalam permintaan integrasi untuk integrasi proxy dan non-proxy.

Sumber data yang dipetakan	Ekspresi pemetaan
Jalur permintaan metode	<code>method.request.path.</code> <i>name</i>
String kueri permintaan metode	<code>method.request.querystring.</code> <i>name</i>
Multi-value string kueri permintaan metode	<code>method.request.multivaluequerystring.</code> <i>name</i>
Header permintaan metode	<code>method.request.header.</code> <i>name</i>
Multi-value header permintaan metode	<code>method.request.multivalueheader.</code> <i>name</i>
Badan permintaan metode	<code>method.request.body</code>
Metode permintaan badan (JsonPath)	<code>method.request.body.</code> <i>JSONPath_EXPRESSION</i> . <i>JSONPath_EXPRESSION</i> adalah ekspresi JSONPath untuk bidang JSON dari badan permintaan. Untuk informasi selengkapnya, lihat ekspresi JsonPath .
Variabel tahap	<code>stageVariables.</code> <i>name</i>
Variabel konteks	<code>context.</code> <i>name</i>

Sumber data yang dipetakan	Ekspresi pemetaan
	Nama harus menjadi salah satu variabel konteks yang didukung .
Nilai statis	<i>'static_value'</i> . <i>static_value</i> Ini adalah string literal dan harus diapit dalam sepasang tanda kutip tunggal. Misalnya, 'https://www.example.com' .

Tabel berikut menunjukkan parameter respons integrasi yang dapat Anda petakan dan ekspresi untuk membuat pemetaan. Dalam ekspresi ini, *name* adalah nama parameter respons integrasi. Anda dapat memetakan header respons metode dari header respons integrasi atau badan respons integrasi, variabel \$context, atau nilai statis. Untuk menggunakan pemetaan parameter untuk respons integrasi, Anda memerlukan integrasi non-proxy.

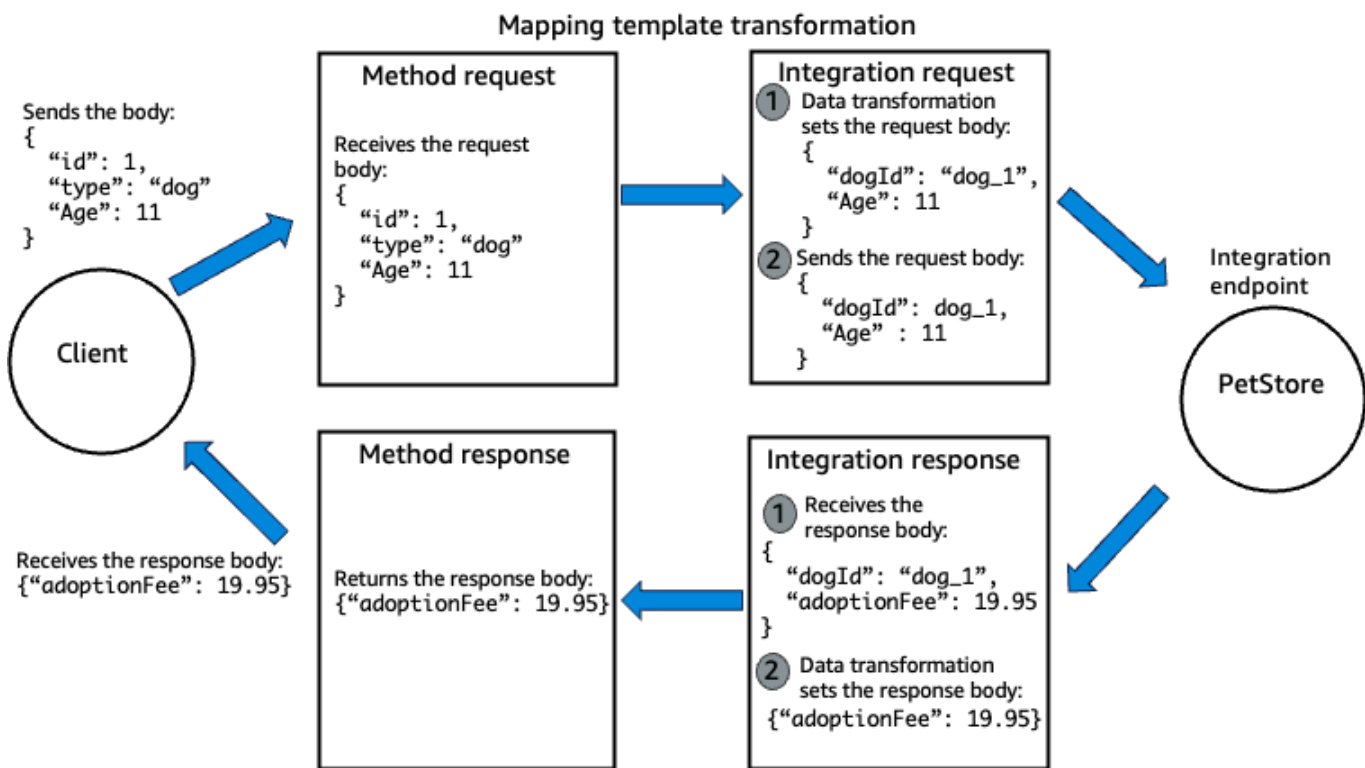
Sumber data yang dipetakan	Ekspresi pemetaan
Header respon integrasi	integration.response.header. <i>name</i>
Header respon integrasi	integration.response.multivalueheader. <i>name</i>
Badan respons integrasi	integration.response.body
Integrasi respon body (JsonPath)	integration.response.body. <i>JSONPath_EXPRESSION</i> <i>JSONPath_EXPRESSION</i> adalah ekspresi JSONPath untuk bidang JSON dari badan respons. Untuk informasi selengkapnya, lihat ekspresi JsonPath.
Variabel tahap	stageVariables. <i>name</i>
Variabel konteks	context. <i>name</i>

Sumber data yang dipetakan	Ekspresi pemetaan
	Nama harus menjadi salah satu variabel konteks yang didukung .
Nilai statis	<i>'static_value'</i> <i>static_value</i> Ini adalah string literal dan harus diapit dalam sepasang tanda kutip tunggal. Misalnya, 'https://www.example.com' .

Memetakan transformasi template untuk REST API di API Gateway

Transformasi template pemetaan menggunakan template pemetaan untuk memodifikasi permintaan integrasi atau respons integrasi Anda. Template pemetaan adalah skrip yang dinyatakan dalam [Velocity Template Language \(VTL\)](#) dan diterapkan ke payload menggunakan [JsonPath](#) berdasarkan header. Content-type Anda menggunakan templat pemetaan saat menggunakan transformasi templat pemetaan. Bagian ini menjelaskan informasi konseptual yang terkait dengan templat pemetaan.

Diagram berikut menunjukkan siklus hidup permintaan untuk POST /pets sumber daya yang memiliki integrasi dengan titik akhir PetStore integrasi. Di API ini, pengguna mengirim data tentang hewan peliharaan dan titik akhir integrasi mengembalikan biaya adopsi yang terkait dengan hewan peliharaan. Dalam siklus hidup permintaan ini, transformasi templat pemetaan memfilter badan permintaan ke titik akhir integrasi dan memfilter badan respons dari titik akhir integrasi.



Bagian berikut menjelaskan siklus hidup permintaan dan respons.

Permintaan metode dan permintaan integrasi

Pada contoh sebelumnya, jika ini adalah badan permintaan yang dikirim ke permintaan metode:

```

POST /pets
HTTP/1.1
Host:abcd1234.us-west-2.amazonaws.com
Content-type: application/json

{
  "id": 1,
  "type": "dog",
  "Age": 11,
}
  
```

Badan permintaan ini tidak dalam format yang benar untuk digunakan oleh titik akhir integrasi, sehingga API Gateway melakukan transformasi template pemetaan. API Gateway hanya melakukan transformasi template pemetaan karena ada template pemetaan yang ditentukan untuk Content-Type `application/json`. Jika Anda tidak mendefinisikan template pemetaan untuk Content-Type,

secara default, API Gateway meneruskan isi melalui permintaan integrasi ke titik akhir integrasi. Untuk mengubah perilaku ini, lihat [the section called “Perilaku permintaan metode untuk muatan tanpa templat pemetaan”](#).

Template pemetaan berikut mengubah data permintaan metode dalam permintaan integrasi sebelum dikirim ke titik akhir integrasi:

```
#set($inputRoot = $input.path('$'))
{
  "dogId" : "dog_"$elem.id,
  "Age": $inputRoot.Age
}
```

1. `$inputRoot` Variabel mewakili objek root dalam data JSON asli dari bagian sebelumnya. Arahkan dimulai dengan `#` simbol.
2. `dogId` adalah gabungan dari pengguna `id` dan nilai string.
3. `Age` berasal dari badan permintaan metode.

Kemudian, output berikut diteruskan ke titik akhir integrasi:

```
{
  "dogId" : "dog_1",
  "Age": 11
}
```

Respon integrasi dan respons metode

Setelah permintaan berhasil ke titik akhir integrasi, titik akhir mengirimkan respons ke respons integrasi API Gateway. Berikut ini adalah contoh data output dari endpoint integrasi:

```
{
  "dogId" : "dog_1",
  "adoptionFee": 19.95,
}
```

Respons metode mengharapkan muatan yang berbeda dari apa yang dikembalikan oleh respons integrasi. API Gateway melakukan transformasi template pemetaan. API Gateway hanya melakukan transformasi template pemetaan karena ada template pemetaan yang ditentukan untuk `Content-Type application/json`. Jika Anda tidak mendefinisikan template pemetaan untuk `Content-Type`,

secara default, API Gateway meneruskan isi melalui respons integrasi ke respons metode. Untuk mengubah perilaku ini, lihat [the section called “Perilaku permintaan metode untuk muatan tanpa templat pemetaan”](#).

```
#set($inputRoot = $input.path('$'))
{
  "adoptionFee" : $inputRoot.adoptionFee,
}
```

Output berikut dikirim ke respon metode:

```
{"adoptionFee": 19.95}
```

Ini melengkapi contoh transformasi template pemetaan. Sebaiknya jika memungkinkan, alih-alih menggunakan transformasi templat pemetaan, Anda menggunakan integrasi proxy untuk mengubah data Anda. Untuk informasi selengkapnya, lihat [the section called “Pilih jenis integrasi API”](#).

Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway

Jika permintaan metode Anda memiliki payload dan Anda tidak memiliki template pemetaan yang ditentukan untuk Content-Type header, Anda dapat memilih untuk meneruskan payload permintaan yang disediakan klien melalui permintaan integrasi ke backend tanpa transformasi. Proses ini dikenal sebagai integrasi passthrough.

Perilaku passthrough aktual dari permintaan yang masuk ditentukan oleh pengaturan ini. Ada tiga opsi:

Bila tidak ada template yang cocok dengan Content-Type header permintaan

Pilih opsi ini jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika jenis konten permintaan metode tidak cocok dengan jenis konten apa pun yang terkait dengan templat pemetaan.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_MATCH` sebagai nilai `passthroughBehavior` properti pada [Integrasi](#).

Ketika tidak ada templat yang ditentukan (disarankan)

Pilih opsi ini jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika tidak ada templat pemetaan yang ditentukan dalam permintaan

integrasi. Jika templat ditentukan saat opsi ini dipilih, permintaan metode dengan jenis muatan dan konten yang tidak cocok dengan templat pemetaan yang ditentukan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_TEMPLATES` sebagai nilai `passthroughBehavior` properti pada [Integrasi](#).

Tidak pernah

Pilih opsi ini jika Anda tidak ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi. Jika templat ditentukan saat opsi ini dipilih, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `NEVER` sebagai nilai `passthroughBehavior` properti pada [Integrasi](#).

Contoh berikut menunjukkan kemungkinan perilaku `passthrough`.

Contoh 1: Satu template pemetaan didefinisikan dalam permintaan integrasi untuk jenis `application/json` konten.

Content-type	Opsi passthrough	Perilaku
Tidak ada API Gateway default ke <code>application/json</code>	<code>WHEN_NO_MATCH</code>	Muatan permintaan diubah menggunakan templat.
Tidak ada API Gateway default ke <code>application/json</code>	<code>WHEN_NO_TEMPLATES</code>	Muatan permintaan diubah menggunakan templat.
Tidak ada API Gateway default ke <code>application/json</code>	<code>NEVER</code>	Muatan permintaan diubah menggunakan templat.

Content-type	Opsi passthrough	Perilaku
application/json	WHEN_NO_MATCH	Muatan permintaan diubah menggunakan templat.
application/json	WHEN_NO_TEMPLATES	Muatan permintaan diubah menggunakan templat.
application/json	NEVER	Muatan permintaan diubah menggunakan templat.
application/xml	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/xml	WHEN_NO_TEMPLATES	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/xml	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.

Contoh 2: Satu template pemetaan didefinisikan dalam permintaan integrasi untuk jenis `application/xml` konten.

Content-type	Opsi passthrough	Perilaku
Tidak ada API Gateway default ke <code>application/json</code>	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
Tidak ada API Gateway default ke <code>application/json</code>	WHEN_NO_TEMPLATES	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.

Content-type	Opsi passthrough	Perilaku
Tidak ada API Gateway default ke application/json	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/json	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/json	WHEN_NO_TEMPLATES	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/json	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/xml	WHEN_NO_MATCH	Muatan permintaan diubah menggunakan templat.
application/xml	WHEN_NO_TEMPLATES	Muatan permintaan diubah menggunakan templat.
application/xml	NEVER	Muatan permintaan diubah menggunakan templat.

Contoh 3: Tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi.

Content-type	Opsi passthrough	Perilaku
Tidak ada API Gateway default ke application/json	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.

Content-type	Opsi passthrough	Perilaku
Tidak ada API Gateway default ke application/json	WHEN_NO_TEMPLATES	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
Tidak ada API Gateway default ke application/json	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/json	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/json	WHEN_NO_TEMPLATES	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/json	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.
application/xml	WHEN_NO_MATCH	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/xml	WHEN_NO_TEMPLATES	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.
application/xml	NEVER	Permintaan ditolak dengan 415 Unsupported Media Type respons HTTP.

Contoh template pemetaan tambahan untuk REST API di API Gateway

Contoh berikut menunjukkan API album foto di API Gateway yang menggunakan templat pemetaan untuk mengubah permintaan integrasi dan data respons integrasi. Ini juga menggunakan model data untuk menentukan permintaan metode dan muatan respons integrasi. Untuk informasi selengkapnya tentang model data, lihat [Model data untuk REST APIs](#).

Permintaan metode dan permintaan integrasi

Berikut ini adalah model yang mendefinisikan badan permintaan metode. Model input ini mengharuskan penelepon mengunggah satu halaman foto, dan membutuhkan minimal 10 foto untuk setiap halaman. Anda dapat menggunakan model input ini untuk menghasilkan SDK atau menggunakan validasi permintaan untuk API Anda. Saat menggunakan validasi permintaan, jika badan permintaan metode tidak mematuhi struktur data model, API Gateway gagal permintaan.

```
{
  "$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" }
    }
}
}
}
}
}
}
}

```

Berikut ini adalah contoh badan permintaan metode yang mematuhi struktur data dari model data sebelumnya.

```

{
  "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
      }
    ]
  }
}

```

Dalam contoh ini, jika badan permintaan metode sebelumnya dikirimkan oleh klien, maka template pemetaan ini mengubah payload agar sesuai dengan format yang diperlukan oleh titik akhir integrasi.

```

#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
  ]
}

```

Contoh berikut adalah data output dari transformasi:

```

{
  "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
  }
]
```

Data ini dikirim ke permintaan integrasi, dan kemudian ke titik akhir integrasi.

Respon integrasi dan respons metode

Berikut ini adalah contoh model keluaran untuk data foto dari titik akhir integrasi. Anda dapat menggunakan model ini untuk model respons metode, yang diperlukan saat Anda membuat SDK yang diketik kuat untuk API. Hal ini menyebabkan output dilemparkan ke kelas yang sesuai di Jawa atau 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" }
        }
      }
    }
  }
}
```

Titik akhir integrasi mungkin tidak merespons dengan respons yang mematuhi struktur data model ini. Misalnya, respons integrasi mungkin terlihat seperti berikut:

```
"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  
  }  
]  
}
```

Contoh template pemetaan berikut mengubah data respons integrasi ke dalam format yang diharapkan oleh respons metode:

```
#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  
  ]  
}
```

Contoh berikut adalah data output dari transformasi:

```
{
  "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
    }
  ]
}
```

Data ini dikirim ke respons metode dan kemudian kembali ke klien.

Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway

Anda dapat menggunakan transformasi templat pemetaan untuk mengganti semua jenis parameter permintaan, header respons, atau kode status respons. Anda menggunakan template pemetaan untuk melakukan hal berikut:

- Lakukan pemetaan parameter banyak-ke-satu
- Ganti parameter setelah pemetaan API Gateway standar diterapkan
- Parameter peta kondisional berdasarkan konten tubuh atau nilai parameter lainnya
- Buat parameter baru secara terprogram
- Ganti kode status yang dikembalikan oleh titik akhir integrasi Anda

Override adalah final. Override hanya dapat diterapkan ke setiap parameter satu kali. Jika Anda mencoba mengganti parameter yang sama beberapa kali, API Gateway mengembalikan 5XX

respons. Jika Anda harus mengganti parameter yang sama beberapa kali di seluruh template, kami sarankan membuat variabel dan menerapkan override di akhir template. Template diterapkan hanya setelah seluruh template diurai.

Contoh 1: Ganti kode status berdasarkan badan integrasi

Contoh berikut menggunakan [API contoh](#) untuk mengganti kode status berdasarkan badan respons integrasi.

Konsol Manajemen AWS

Untuk mengganti kode status berdasarkan badan respons integrasi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API.
3. Untuk REST API, pilih Build.
4. Untuk detail API, pilih API Contoh.
5. Pilih Buat API.

API Gateway membuat contoh API toko hewan peliharaan. Untuk mengambil informasi tentang hewan peliharaan, Anda menggunakan permintaan metode APIGET `/pets/{petId}`, di mana `{petId}` parameter jalur yang sesuai dengan nomor ID untuk hewan peliharaan.

Dalam contoh ini, Anda mengganti kode respons GET metode 400 saat kondisi kesalahan terdeteksi.

6. Di pohon Resources, pilih GET metode di bawah `{petId}`.
7. Pertama, Anda menguji implementasi API saat ini.

Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

8. Untuk PetID, masukkan **-1**, lalu pilih Uji.

Badan Respons menunjukkan kesalahan di luar jangkauan:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
```

```

    "message": "The value is out of range."
  }
]
}

```

Selain itu, baris terakhir di bawah Log diakhiri dengan: `Method completed with status: 200`.

Integrasi berhasil diselesaikan, tetapi ada kesalahan. Sekarang Anda akan mengganti kode status berdasarkan respons integrasi.

9. Pada tab Respons Integrasi, untuk Default - Respons, pilih Edit.
10. Pilih template Pemetaan.
11. Pilih Tambahkan templat pemetaan.
12. Untuk jenis Konten, masukkan **application/json**.
13. Untuk badan Template, masukkan yang berikut ini:

```

#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end

```

Template pemetaan ini menggunakan `$context.responseOverride.status` variabel untuk mengganti kode status 400 jika respons integrasi berisi string. `error`

14. Pilih Simpan.
15. Pilih tab Uji.
16. Untuk PeTiD, masukkan. **-1**
17. Dalam hasilnya, Badan Respons menunjukkan kesalahan di luar jangkauan:

```

{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}

```

Namun, baris terakhir di bawah Log sekarang berakhir dengan:Method completed with status: 400.

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

```
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/
                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

Definisi OpenAPI berikut membuat GET `pets/{petId}` sumber daya dan mengganti kode status berdasarkan badan integrasi.

```

{
  "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\n$input.json(\"$\")\n\n#if($inputRoot.toString().contains(\"error\n\"))\n\n#set($context.responseOverride.status = 400)\n\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"
      }
    }
  }
}
```

Contoh 2: Ganti header permintaan dan buat header baru

Contoh berikut menggunakan [API contoh](#) untuk mengganti header permintaan dan membuat header baru.

Konsol Manajemen AWS

Untuk mengganti header permintaan metode dengan membuat header baru

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih contoh API yang Anda buat di tutorial sebelumnya. Nama API seharusnya PetStore.
3. Di pohon Resources, pilih GET metode di bawah/pet.
4. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
5. Pilih header permintaan HTTP, lalu pilih Tambah header.
6. Untuk Nama, masukkan **header1**.
7. Pilih Tambahkan header, lalu buat header kedua yang disebut **header2**.
8. Pilih Simpan.

Sekarang, Anda menggabungkan header ini menjadi satu nilai header menggunakan template pemetaan.

9. Pada tab Permintaan integrasi, untuk pengaturan permintaan Integrasi, pilih Edit.
10. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
11. Pilih template Pemetaan, lalu lakukan hal berikut:
 - a. Pilih Tambahkan templat pemetaan.
 - b. Untuk jenis Konten, masukkan **application/json**.
 - c. Untuk badan Template, masukkan yang berikut ini:

```
#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])
```

Template pemetaan ini diganti header1 dengan string pets dan membuat header multi-nilai yang disebut \$header3Value yang menggabungkan dan. header1 header2

12. Pilih Simpan.
13. Pilih tab Uji.
14. Di bawah Header, salin kode berikut:

```
header1:header1Val
header2:header2Val
```

15. Pilih Uji.

Di Log, Anda akan melihat entri yang menyertakan teks ini:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=pets, x-amzn-apigateway-api-id=api-id,
Accept=application/json, multivalueheader=pets,header1Valheader2Val}
```

CloudFormation

Dalam contoh ini, Anda menggunakan properti [body](#) untuk mengimpor file definisi OpenAPI ke API Gateway.

```

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

Definisi OpenAPI berikut membuat GET `/pets` sumber daya dan mengganti header permintaan dan membuat header baru.

```
{
  "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"
        }
    }
}
}
}
}

```

Untuk menggunakan penggantian template pemetaan, tambahkan satu atau beberapa variabel berikut `$context`. Untuk daftar `$context` variabel, lihat [the section called “Variabel konteks untuk transformasi data”](#).

Tutorial: Memodifikasi permintaan integrasi dan respon untuk integrasi AWS layanan

Tutorial berikut menunjukkan cara menggunakan transformasi template pemetaan untuk mengatur template pemetaan untuk mengubah permintaan integrasi dan tanggapan menggunakan konsol dan CLI. AWS

Topik

- [Mengatur transformasi data menggunakan konsol API Gateway](#)
- [Mengatur transformasi data menggunakan AWS CLI](#)
- [Transformasi data yang lengkap CloudFormation templat](#)

Mengatur transformasi data menggunakan konsol API Gateway

[Dalam tutorial ini, Anda akan membuat tabel API dan DynamoDB yang tidak lengkap menggunakan file.zip berikut data-transformation-tutorial-console.zip.](#) API yang tidak lengkap ini memiliki /pets sumber daya dengan GET dan POST metode.

- GETMetode ini akan mendapatkan data dari titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Data keluaran akan diubah sesuai dengan template pemetaan di [the section called “Pemetaan transformasi template”](#).
- POSTMetode ini akan memungkinkan pengguna untuk menyimpan informasi POST ke tabel Amazon DynamoDB menggunakan template pemetaan.

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#). Anda akan menggunakan template ini untuk membuat tabel DynamoDB untuk memposting informasi hewan peliharaan dan API yang tidak lengkap. Anda akan menyelesaikan langkah-langkah lainnya di konsol API Gateway.

Untuk membuat CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, **data-transformation-tutorial-console** masukkan lalu pilih Berikutnya.

7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Berikutnya, lalu pilih Kirim.

CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Untuk menguji respons integrasi **GET**

1. Pada tab Sumber Daya CloudFormation tumpukan untuk **data-transformation-tutorial-console**, pilih ID fisik API Anda.
2. Di panel navigasi utama, pilih Resources, lalu pilih metode GET.
3. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

Output dari tes akan menunjukkan yang berikut:

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

Anda akan mengubah output ini sesuai dengan template pemetaan di [the section called “Pemetaan transformasi template”](#).

Untuk mengubah respons integrasi **GET**

1. Pilih tab Respons Integrasi.

Saat ini, tidak ada template pemetaan yang ditentukan, sehingga respons integrasi tidak akan diubah.

2. Untuk Default - Respons, pilih Edit.

3. Pilih template Pemetaan, lalu lakukan hal berikut:

- Pilih Tambahkan templat pemetaan.
- Untuk jenis Konten, masukkan **application/json**.
- Untuk badan Template, masukkan yang berikut ini:

```
#set($inputRoot = $input.path('$'))
[
  #foreach($elem in $inputRoot)
    {
      "description" : "Item $elem.id is a $elem.type.",
      "askingPrice" : $elem.price
    }#if($foreach.hasNext),#end
  #end
]
```

Pilih Simpan.

Untuk menguji respons integrasi **GET**

- Pilih tab Test, lalu pilih Test.

Output dari tes akan menunjukkan respons yang ditransformasikan.

```
[
  {
    "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
  }
]
```

Untuk mengubah data input dari metode **POST**

1. Pilih metode POST.
2. Pilih tab Permintaan integrasi, dan kemudian untuk pengaturan permintaan Integrasi, pilih Edit.

CloudFormation Template telah mengisi beberapa bidang permintaan integrasi.

- Jenis integrasi adalah Layanan AWS.
- Layanan AWS Ini adalah DynamoDB.
- Metode HTTP adalah POST.
- Tindakan adalah PutItem.
- Peran Eksekusi yang memungkinkan API Gateway untuk menempatkan item ke dalam tabel DynamoDB adalah `data-transformation-tutorial-console-APIGatewayRole`. CloudFormation membuat peran ini untuk memungkinkan API Gateway memiliki izin minimal untuk berinteraksi dengan DynamoDB.

Nama tabel DynamoDB belum ditentukan. Anda akan menentukan nama dalam langkah-langkah berikut.

3. Untuk Request body passthrough, pilih Never.

Ini berarti bahwa API akan menolak data dengan Content-Types yang tidak memiliki template pemetaan.

4. Pilih template Pemetaan.
5. Jenis Konten diatur ke `application/json`. Ini berarti jenis konten yang tidak `application/json` akan ditolak oleh API. Untuk informasi selengkapnya tentang perilaku passthrough integrasi, lihat [Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway](#)
6. Masukkan kode berikut ke dalam editor teks.

```
{
  "TableName": "data-transformation-tutorial-console-ddb",
  "Item": {
    "id": {
      "N": $input.json("$.id")
    },
    "type": {
      "S": $input.json("$.type")
    },
    "price": {
      "N": $input.json("$.price")
    }
  }
}
```

Template ini menentukan tabel sebagai `data-transformation-tutorial-console-ddb` dan menetapkan item sebagai `id`, `type`, dan `price`. Item akan berasal dari tubuh POST metode. Anda juga dapat menggunakan model data untuk membantu membuat template pemetaan. Untuk informasi selengkapnya, lihat [Minta validasi untuk REST APIs di API Gateway](#).

7. Pilih Simpan untuk menyimpan template pemetaan Anda.

Untuk menambahkan metode dan respons integrasi dari metode **POST**

CloudFormation Membuat metode kosong dan respons integrasi. Anda akan mengedit tanggapan ini untuk memberikan informasi lebih lanjut. Untuk informasi selengkapnya tentang cara mengedit tanggapan, lihat [Contoh pemetaan parameter untuk REST API di API Gateway](#).

1. Pada tab Respons Integrasi, untuk Default - Respons, pilih Edit.
2. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan.
3. Untuk Content-type, masukkan **application/json**.
4. Di editor kode, masukkan template pemetaan keluaran berikut untuk mengirim pesan output:

```
{ "message" : "Your response was recorded at $context.requestTime" }
```

Untuk informasi lebih lanjut tentang variabel konteks, lihat [Variabel konteks untuk transformasi data](#).

5. Pilih Simpan untuk menyimpan template pemetaan Anda.

Uji metode **POST**

Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

1. Di badan permintaan, masukkan contoh berikut.

```
{
    "id": "4",
    "type" : "dog",
    "price": "321"
}
```

2. Pilih Uji.

Output harus menunjukkan pesan sukses Anda.

Anda dapat membuka konsol DynamoDB <https://console.aws.amazon.com/dynamodb/> di untuk memverifikasi bahwa item contoh ada di tabel Anda.

Untuk menghapus CloudFormation tumpukan

1. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.
2. Pilih CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasi pilihan Anda.

Mengatur transformasi data menggunakan AWS CLI

Dalam tutorial ini, Anda akan membuat tabel API dan DynamoDB yang tidak lengkap menggunakan file.zip berikut [data-transformation-tutorial-cli.zip](#). API yang tidak lengkap ini memiliki /pets sumber daya dengan GET metode yang terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan membuat POST metode untuk terhubung ke tabel DynamoDB dan menggunakan template pemetaan untuk memasukkan data ke dalam tabel DynamoDB.

- Anda akan mengubah data keluaran sesuai dengan template pemetaan di [the section called “Pemetaan transformasi template”](#).
- Anda akan membuat POST metode untuk memungkinkan pengguna untuk menyimpan informasi ke POST tabel Amazon DynamoDB menggunakan template pemetaan.

Untuk membuat CloudFormation tumpukan

Unduh dan unzip [template pembuatan aplikasi untuk CloudFormation](#).

Untuk menyelesaikan tutorial berikut, Anda memerlukan [AWS Command Line Interface \(AWS CLI\) versi 2](#).

Untuk perintah panjang, karakter escape (\) digunakan untuk memisahkan perintah menjadi beberapa baris.

 Note

Di Windows, beberapa perintah Bash CLI yang biasa Anda gunakan (zipseperti) tidak didukung oleh terminal bawaan sistem operasi. Untuk mendapatkan Windows-integrated versi Ubuntu dan Bash, [instal Subsistem Windows untuk Linux](#). Contoh perintah CLI dalam panduan ini menggunakan pemformatan Linux. Perintah yang menyertakan dokumen JSON sebaris harus diformat ulang jika Anda menggunakan CLI Windows.

1. Gunakan perintah berikut untuk membuat CloudFormation tumpukan.

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
--template-body file://data-transformation-tutorial-cli.zip --capabilities
CAPABILITY_NAMED_IAM
```

2. CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Gunakan perintah berikut untuk melihat status CloudFormation tumpukan Anda.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Ketika status CloudFormation tumpukan Anda `StackStatus: "CREATE_COMPLETE"`, gunakan perintah berikut untuk mengambil nilai output yang relevan untuk langkah-langkah masa depan.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

Nilai output adalah sebagai berikut:

- `ApiRole`, yang merupakan nama peran yang memungkinkan API Gateway untuk menempatkan item dalam tabel DynamoDB. Untuk tutorial ini, nama perannya adalah `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFGH`.
- `DDBTableName`, yang merupakan nama tabel DynamoDB. Untuk tutorial ini, nama tabelnya adalah `data-transformation-tutorial-cli-ddb`
- `ResourceId`, yang merupakan ID untuk sumber daya hewan peliharaan tempat POST metode GET dan diekspos. Untuk tutorial ini, Resource ID adalah `efg456`
- `ApiId`, yang merupakan ID untuk API. Untuk tutorial ini, ID API adalah `abc123`.

Untuk menguji metode **GET** sebelum transformasi data

- Gunakan perintah berikut untuk menguji GET metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET
```

Output dari tes akan menunjukkan yang berikut ini.

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

Anda akan mengubah output ini sesuai dengan template pemetaan di [the section called “Pemetaan transformasi template”](#).

Untuk mengubah respons integrasi **GET**

- Gunakan perintah berikut untuk memperbarui respons integrasi untuk GET metode ini. Ganti *rest-api-id* dan *resource-id* dengan nilai-nilai Anda.

Gunakan perintah berikut untuk membuat respons integrasi.

```
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]"]'}
```

Untuk menguji metode **GET**

- Gunakan perintah berikut untuk menguji GET metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

Output dari tes akan menunjukkan respons yang ditransformasikan.

```
[
  {
    "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
}
```

Untuk membuat metode **POST**

1. Gunakan perintah berikut untuk membuat metode baru pada /pets sumber daya.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Metode ini akan memungkinkan Anda untuk mengirim informasi hewan peliharaan ke tabel DynamoDB yang Anda buat di tumpukan. CloudFormation

2. Gunakan perintah berikut untuk membuat Layanan AWS integrasi pada POST metode.

```
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. Gunakan perintah berikut untuk membuat respons metode untuk panggilan POST metode yang berhasil.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

- Gunakan perintah berikut untuk membuat respons integrasi untuk panggilan POST metode yang berhasil.

```
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\\"}"}'
```

Untuk menguji metode **POST**

- Gunakan perintah berikut untuk menguji POST metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{"id": "4", "type": "dog", "price": "321"}'
```

Output akan menampilkan pesan yang berhasil.

Untuk menghapus CloudFormation tumpukan

- Gunakan perintah berikut untuk menghapus CloudFormation sumber daya Anda.

```
aws cloudformation delete-stack --stack-name data-transformation-tutorial-cli
```

Transformasi data yang lengkap CloudFormation templat

Contoh berikut adalah CloudFormation template lengkap, yang membuat API dan tabel DynamoDB dengan sumber daya /pets GET dengan dan metode. POST

- GETMetode ini akan mendapatkan data dari titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Data keluaran akan diubah sesuai dengan template pemetaan di [the section called “Pemetaan transformasi template”](#).
- POSTMetode ini akan memungkinkan pengguna untuk menyimpan informasi POST ke tabel DynamoDB menggunakan template pemetaan.

Contoh CloudFormation templat

```
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

```

Contoh menggunakan variabel untuk memetakan transformasi template untuk API Gateway

Contoh berikut menunjukkan cara menggunakan `$context`, `input`, dan `util` variabel dalam template pemetaan. Anda dapat menggunakan integrasi tiruan atau integrasi non-proxy Lambda

yang mengembalikan peristiwa input kembali ke API Gateway. Untuk daftar semua variabel yang didukung untuk transformasi data, lihat [the section called “Variabel untuk transformasi data”](#).

Contoh 1: Berikan beberapa variabel **\$context** ke titik akhir integrasi

Contoh berikut menunjukkan template pemetaan yang memetakan \$context variabel masuk ke variabel backend dengan nama yang sedikit berbeda dalam payload permintaan integrasi:

```
{
  "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"
}
```

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{
  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'
}
```

Salah satu variabelnya adalah kunci API. Contoh ini mengasumsikan bahwa metode tersebut memerlukan kunci API.

Contoh 2: Teruskan semua parameter permintaan ke titik akhir integrasi melalui payload JSON

Contoh berikut meneruskan semua parameter permintaan, termasuk `path`, `queryString`, dan header parameter, ke titik akhir integrasi melalui payload JSON:

```
#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
  }
}
```

Jika permintaan memiliki parameter input berikut:

- Parameter jalur bernama `myparam`
- Parameter string kueri `queryString1=value1,value2`
- Header. `"header1" : "value1"`

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{"params":{"path":{"example2":"myparamm"},"queryString":
{"queryString1":"value1,value2"},"header":{"header1":"value1"}}
```

Contoh 3: Lulus subbagian dari permintaan metode ke titik akhir integrasi

Contoh berikut menggunakan parameter input `name` untuk mengambil hanya `name` parameter dan parameter input `input.json('$')` untuk mengambil seluruh isi permintaan metode:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('$')
}
```

Untuk permintaan yang menyertakan parameter string kueri `name=Bella&type=dog` dan isi berikut:

```
{
  "Price" : "249.99",
  "Age": "6"
}
```

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{
  "name" : "Bella",
  "body" : {"Price":"249.99","Age":"6"}
}
```

Template pemetaan ini menghapus parameter `type=dog` string kueri.

Jika input JSON berisi karakter unescaped yang tidak dapat diuraikan, API JavaScript Gateway mungkin menampilkan respons 400. Terapkan `$util.escapeJavaScript($input.json('$'))` untuk memastikan input JSON dapat diurai dengan benar.

Contoh sebelumnya dengan `$util.escapeJavaScript($input.json('$'))` diterapkan adalah sebagai berikut:

```
{
  "name" : "$input.params('name')",
  "body" : "$util.escapeJavaScript($input.json('$'))"
}
```

Dalam hal ini, output dari template pemetaan ini akan terlihat seperti berikut:

```
{
  "name" : "Bella",
  "body": {"Price":"249.99","Age":"6"}
}
```

Contoh 4: Gunakan ekspresi JsonPath untuk meneruskan subbagian dari permintaan metode ke titik akhir integrasi

Contoh berikut menggunakan ekspresi JsonPath untuk mengambil hanya parameter input name dan Age dari badan permintaan:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('$.Age')
}
```

Untuk permintaan yang menyertakan parameter string kueri name=Bella&type=dog dan isi berikut:

```
{
  "Price" : "249.99",
  "Age": "6"
}
```

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{
  "name" : "Bella",
  "body" : "6"
}
```

Template pemetaan ini menghapus parameter string kueri type=dog dan Price bidang dari badan.

Jika payload permintaan metode berisi karakter unescaped yang tidak dapat diuraikan, API JavaScript Gateway mungkin menampilkan respons. 400 Terapkan `$util.escapeJavaScript()` untuk memastikan input JSON dapat diurai dengan benar.

Contoh sebelumnya dengan `$util.escapeJavaScript($input.json('$.Age'))` diterapkan adalah sebagai berikut:

```
{
  "name" : "$input.params('name')",
  "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

Dalam hal ini, output dari template pemetaan ini akan terlihat seperti berikut:

```
{
  "name" : "Bella",
  "body": "\"6\""
}
```

Contoh 5: Gunakan ekspresi JsonPath untuk meneruskan informasi tentang permintaan metode ke titik akhir integrasi

Contoh berikut menggunakan `$input.params()`, `$input.path()`, dan `$input.json()` untuk mengirim informasi tentang permintaan metode ke titik akhir integrasi. Template pemetaan ini menggunakan `size()` metode untuk memberikan jumlah elemen dalam daftar.

```
{
  "id" : "$input.params('id')",
  "count" : "$input.path('$.things').size()",
  "things" : $input.json('$.things')
}
```

Untuk permintaan yang menyertakan parameter jalur 123 dan isi berikut:

```
{
  "things": {
    "1": {},
    "2": {},
    "3": {}
  }
}
```

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Jika payload permintaan metode berisi karakter unescaped yang tidak dapat diuraikan, API JavaScript Gateway mungkin menampilkan respons. 400 Terapkan `$util.escapeJavaScript()` untuk memastikan input JSON dapat diurai dengan benar.

Contoh sebelumnya dengan `$util.escapeJavaScript($input.json('$.things'))` diterapkan adalah sebagai berikut:

```
{
```

```
"id" : "$input.params('id')",
"count" : "$input.path('$.things').size()",
"things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

Output dari template pemetaan ini akan terlihat seperti berikut:

```
{"id":"123","count":"3","things":{"\"1\":{},\"2\":{},\"3\":{}}}
```

Variabel untuk transformasi data untuk API Gateway

Saat Anda membuat pemetaan parameter, Anda dapat menggunakan variabel konteks sebagai sumber data Anda. Saat Anda membuat transformasi template pemetaan, Anda dapat menggunakan variabel konteks, input, dan variabel util dalam skrip yang Anda tulis di [Velocity Template Language \(VTL\)](#). Misalnya template pemetaan yang menggunakan variabel referensi ini, lihat [the section called “Contoh menggunakan variabel untuk pemetaan transformasi template”](#).

Untuk daftar variabel referensi untuk pencatatan akses, lihat [the section called “Variabel untuk pencatatan akses untuk API Gateway”](#).

Variabel konteks untuk transformasi data

Anda dapat menggunakan `$context` variabel case-sensitive berikut untuk transformasi data.

Parameter	Deskripsi
<code>\$context.accountId</code>	ID AWS akun pemilik API.
<code>\$context.apiId</code>	API Gateway identifier ditetapkan ke API Anda.
<code>\$context.authorizer.claims. <i>property</i></code>	Properti klaim yang dikembalikan dari kumpulan pengguna Amazon Cognito setelah pemanggil metode berhasil diautentikasi. Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API” .

Parameter	Deskripsi
	 Note Memanggil <code>\$context.authorizer.claims</code> mengembalikan null.
<code>\$context.authorizer.principalId</code>	Identifikasi pengguna utama yang terkait dengan token yang dikirim oleh klien dan dikembalikan dari otorisasi API Gateway Lambda (sebelumnya dikenal sebagai otorisasi khusus). Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda” .

Parameter	Deskripsi
<code>\$context.authorizer.</code> <i>property</i>	Nilai stringifikasi dari pasangan nilai kunci <code>context</code> peta yang ditentukan dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan <code>context</code> peta berikut: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Memanggil <code>\$context.authorizer.key</code> mengembalikan "value" string, memanggil <code>\$context.authorizer.numKey</code> mengembalikan "1" string, dan memanggil <code>\$context.authorizer.boolKey</code> mengembalikan "true" string. Sebab <i>property</i>, satu-satunya karakter khusus yang didukung adalah (<code>_</code>) karakter garis bawah. Untuk informasi selengkapnya, lihat the section called "Gunakan otorisasi Lambda".
<code>\$context.awsEndpointRequestId</code>	ID permintaan AWS titik akhir.
<code>\$context.deploymentId</code>	ID penerapan API.
<code>\$context.domainName</code>	Nama domain lengkap yang digunakan untuk memanggil API. Ini harus sama dengan Host header yang masuk.
<code>\$context.domainPrefix</code>	Label pertama dari <code>\$context.domainName</code> .

Parameter	Deskripsi
<code>\$context.error.message</code>	String yang berisi pesan kesalahan API Gateway. Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponse pemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Untuk informasi selengkapnya, lihat the section called “Metrik-metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan” .
<code>\$context.error.messageString</code>	Nilai yang dikutip dari <code>\$context.error.message</code> , yaitu <code>"\$context.error.message"</code> .
<code>\$context.error.responseType</code>	Sebuah jenis GatewayResponse . Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponse pemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Untuk informasi selengkapnya, lihat the section called “Metrik-metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan” .
<code>\$context.error.validationErrorString</code>	Sebuah string yang berisi pesan kesalahan validasi rinci.
<code>\$context.extendedRequestId</code>	ID tambahan yang dibuat dan ditetapkan API Gateway ke permintaan API. ID permintaan yang diperluas berisi informasi yang berguna untuk debugging dan pemecahan masalah.

Parameter	Deskripsi
<code>\$context.httpMethod</code>	Metode HTTP yang digunakan . Nilai yang valid meliputi: DELETE, GET, HEAD, OPTIONS, PATCH, POST, dan PUT.
<code>\$context.identity.accountId</code>	ID AWS akun yang terkait dengan permintaan.
<code>\$context.identity.apiKey</code>	Untuk metode API yang memerlukan kunci API, variabel ini adalah kunci API yang terkait dengan permintaan metode. Untuk metode yang tidak memerlukan kunci API, variabel ini adalah null. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan” .
<code>\$context.identity.apiKeyId</code>	ID kunci API yang terkait dengan permintaan API yang memerlukan kunci API.
<code>\$context.identity.caller</code>	Pengenal utama penelepon yang menandatangani permintaan. Didukung untuk sumber daya yang menggunakan otorisasi IAM.

Parameter	Deskripsi
<code>\$context.identity.cognitoAuthenticationProvider</code>	Daftar dipisahkan koma dari semua penyedia otentikasi Amazon Cognito yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Untuk informasi tentang penyedia autentikasi Amazon Cognito yang tersedia, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Nilai yang mungkin termasuk <code>authenticated</code> untuk identitas yang diautentikasi dan <code>unauthenticated</code> untuk identitas yang tidak diautentikasi.
<code>\$context.identity.cognitoIdentityId</code>	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.

Parameter	Deskripsi
<code>\$context.identity.principalOrgId</code>	ID AWS organisasi .
<code>\$context.identity.sourceIp</code>	Alamat IP sumber dari koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Sertifikat PEM-encoded klien yang disajikan klien selama otentikasi TLS timbal balik. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
<code>\$context.identity.clientCertificate.subjectDN</code>	Nama yang dibedakan dari subjek sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
<code>\$context.identity.clientCertificate.issuerDN</code>	Nama terkemuka penerbit sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
<code>\$context.identity.clientCertificate.serialNumber</code>	Nomor seri sertifikat. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.

Parameter	Deskripsi
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Tanggal sebelum sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Tanggal setelah sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
<code>\$context.identity.vpcId</code>	ID VPC VPC membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.vpceId</code>	ID titik akhir VPC dari titik akhir VPC membuat permintaan ke titik akhir API Gateway. Hadir hanya ketika Anda memiliki API pribadi.
<code>\$context.identity.user</code>	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk sumber daya yang menggunakan otorisasi IAM.
<code>\$context.identity.userAgent</code>	User-Agent Header pemanggil API.
<code>\$context.identity.userArn</code>	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.isCanaryRequest</code>	Kembali <code>true</code> jika permintaan diarahkan ke kenari dan <code>false</code> jika permintaan tidak diarahkan ke kenari. Hadir hanya ketika Anda mengaktifkan kenari.

Parameter	Deskripsi
<code>\$context.path</code>	Jalur permintaan. Misalnya, untuk URL permintaan non-proxy dari <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , <code>\$context.path</code> nilainya adalah <code>{stage}/root/child</code> .
<code>\$context.protocol</code>	Protokol permintaan, misalnya, HTTP/1.1.
	 Note API Gateway API dapat menerima HTTP/2 permintaan, tetapi API Gateway mengirimkan permintaan ke integrasi backend menggunakan HTTP/1.1. Akibatnya, protokol permintaan dicatat seolah-olah HTTP/1.1 klien mengirim permintaan yang menggunakan HTTP/2.
<code>\$context.requestId</code>	ID untuk permintaan. Klien dapat mengganti ID permintaan ini. Gunakan <code>\$context.extendedRequestId</code> untuk ID permintaan unik yang dihasilkan API Gateway.
<code>\$context.requestOverride.header. <i>header_name</i></code>	Header permintaan menimpa. Jika parameter ini didefinisikan, ini berisi header yang akan digunakan alih-alih Header HTTP yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .

Parameter	Deskripsi
<code>\$context.requestOverride.path.</code> <i>path_name</i>	Jalur permintaan menerima. Jika parameter ini ditentukan, parameter ini berisi jalur permintaan yang akan digunakan, bukan Parameter Jalur URL yang ditentukan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.requestOverride.querystring.</code> <i>querystring_name</i>	Permintaan query string override. Jika parameter ini didefinisikan, ini berisi string permintaan yang akan digunakan alih-alih Parameter String Kueri URL yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.responseOverride.header.</code> <i>header_name</i>	Header respon menerima. Jika parameter ini didefinisikan, ini berisi header yang akan dikembalikan, bukan header Response yang didefinisikan sebagai pemetaan Default di panel Integration Response. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .

Parameter	Deskripsi
<code>\$context.responseOverride.status</code>	Kode status respons menerima. Jika parameter ini didefinisikan, ini berisi kode status yang akan dikembalikan, bukan status respons Metode yang didefinisikan sebagai pemetaan Default di panel Respons Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.requestTime</code>	Waktu permintaan yang diformat CLF (). dd/MMM/yyyy:HH:mm:ss +-hhmm
<code>\$context.requestTimeEpoch</code>	Waktu permintaan yang diformat Epoch , dalam milidetik.
<code>\$context.resourceId</code>	Pengidentifikasi yang ditetapkan API Gateway ke sumber daya Anda.
<code>\$context.resourcePath</code>	Jalan menuju sumber daya Anda. Misalnya, untuk URI permintaan non-proxy dari <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , <code>\$context.resourcePath</code> Nilainya adalah <code>/root/child</code> . Untuk informasi selengkapnya, lihat Tutorial: Membuat REST API dengan integrasi non-proxy HTTP .
<code>\$context.stage</code>	Tahap penerapan permintaan API (misalnya, Beta atau Prod).
<code>\$context.wafResponseCode</code>	Tanggapan yang diterima dari AWS WAF : <code>WAF_ALLOW</code> atau <code>WAF_BLOCK</code> . Tidak akan diatur jika tahap tidak terkait dengan ACL web. Untuk informasi selengkapnya, lihat the section called “AWS WAF” .

Parameter	Deskripsi
<code>\$context.webaclArn</code>	ARN lengkap dari ACL web yang digunakan untuk memutuskan apakah akan mengizinkan atau memblokir permintaan. Tidak akan diatur jika tahap tidak terkait dengan ACL web. Untuk informasi selengkapnya, lihat the section called “AWS WAF” .

Variabel masukan

Anda dapat menggunakan `$input` variabel case-sensitive berikut untuk merujuk ke metode permintaan payload dan parameter permintaan metode. Fungsi-fungsi berikut tersedia:

Variabel dan fungsi	Deskripsi
<code>\$input.body</code>	Mengembalikan payload permintaan mentah sebagai string. Anda dapat menggunakan <code>\$input.body</code> untuk mempertahankan seluruh nomor floating point, seperti <code>10.00</code> .
<code>\$input.json(x)</code>	Fungsi ini mengevaluasi ekspresi JSONPath dan mengembalikan hasil sebagai string JSON. Misalnya, <code>\$input.json('\$.pets')</code> mengembalikan string JSON yang mewakili <code>pets</code> struktur. Untuk informasi selengkapnya tentang JsonPath, lihat JsonPath atau JsonPath for Java.
<code>\$input.params()</code>	Mengembalikan peta dari semua parameter permintaan. Kami menyarankan Anda menggunakan <code>\$util.escapeJavaScript</code> untuk membersihkan hasilnya untuk menghindari potensi serangan injeksi. Untuk kontrol penuh sanitasi permintaan, gunakan integrasi proxy

Variabel dan fungsi	Deskripsi
	tanpa templat dan tangani sanitasi permintaan dalam integrasi Anda.
<code>\$input.params(x)</code>	Mengembalikan nilai parameter permintaan metode dari path, query string, atau nilai header (dicari dalam urutan itu), diberikan string x nama parameter. Kami menyarankan Anda menggunakan <code>\$util.escapeJavaScript</code> untuk membersihkan parameter untuk menghindari potensi serangan injeksi. Untuk kontrol penuh sanitasi parameter, gunakan integrasi proxy tanpa templat dan tangani sanitasi permintaan dalam integrasi Anda.

Variabel dan fungsi	Deskripsi
<code>\$input.path(x)</code>	Mengambil ekspresi string JSONPath (x) dan mengembalikan representasi objek JSON dari hasil. Ini memungkinkan Anda untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika ekspresi <code>\$input.path('\$.pets')</code> mengembalikan objek seperti ini: <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> akan kembali "3". Untuk informasi selengkapnya tentang JsonPath, lihat JsonPath atau JsonPath for Java.

Variabel tahap

Anda dapat menggunakan variabel tahap berikut sebagai placeholder untuk ARN dan URL dalam integrasi metode. Untuk informasi selengkapnya, lihat [the section called “Gunakan variabel tahap”](#).

Sintaksis	Deskripsi
<code>\$stageVariables. <i>variable_name</i> , \$stageVariables[' <i>variable_name</i> '], atau \${stageVariables[' <i>variable_name</i>']}</code>	<i>variable_name</i> merupakan nama variabel tahap.

Variabel Util

Anda dapat menggunakan `$util` variabel case-sensitive berikut untuk menggunakan fungsi utilitas untuk memetakan template. Kecuali ditentukan lain, set karakter default adalah UTF-8.

Fungsi	Deskripsi
<code>\$util.escapeJavaScript()</code>	Melarikan diri dari karakter dalam string menggunakan aturan JavaScript string.  Note Fungsi ini akan mengubah tanda kutip tunggal biasa (') menjadi yang keluar (\'). Namun, tanda kutip tunggal yang lolos tidak valid di JSON. Jadi, ketika output dari fungsi ini digunakan dalam properti JSON, Anda harus mengubah tanda kutip tunggal yang diloloskan (\') kembali ke tanda kutip tunggal biasa ('). Ini ditunjukkan dalam contoh berikut:

Fungsi	Deskripsi
	<pre>"input" : "\$util.escapeJavaScript(<i>data</i>).replaceAll("\\'", "'")"</pre>
<pre>\$util.parseJson()</pre>	Mengambil “stringified” JSON dan mengembalikan representasi objek dari hasilnya. Anda dapat menggunakan hasil dari fungsi ini untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika Anda memiliki muatan berikut: <pre>{"errorMessage":{"key1":"var1","key2":{"arr":[1,2,3]}}}</pre> dan gunakan template pemetaan berikut <pre>#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$errorMessage'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> Anda akan mendapatkan output sebagai berikut: <pre>{ "errorMessageObjKey2ArrVal" : 1 }</pre>
<pre>\$util.urlEncode()</pre>	Mengkonversi string ke dalam format "application/x-www-form-urlencoded".

Fungsi	Deskripsi
<code>\$util.urlDecode()</code>	Mendekode string "application/x-www-form-urlencoded".
<code>\$util.base64Encode()</code>	Mengkodekan data ke dalam string yang dikodekan base64.
<code>\$util.base64Decode()</code>	Mendekode data dari string yang dikodekan base64.

Tanggapan gateway untuk REST APIs di API Gateway

Respons gateway diidentifikasi oleh tipe respons yang ditentukan oleh API Gateway. [Respons terdiri dari kode status HTTP, satu set header tambahan yang ditentukan oleh pemetaan parameter, dan payload yang dihasilkan oleh template pemetaan non-VTL.](#)

Di API Gateway REST API, respons gateway diwakili oleh [GatewayResponse](#). Di OpenAPI, GatewayResponse instance dijelaskan oleh ekstensi [x-amazon-apigateway-gateway-responses.gatewayResponse](#).

Untuk mengaktifkan respons gateway, Anda menyiapkan respons gateway untuk [jenis respons yang didukung](#) di API level. Setiap kali API Gateway menampilkan respons jenis ini, templat pemetaan header dan pemetaan muatan yang ditentukan dalam respons gateway diterapkan untuk mengembalikan hasil yang dipetakan ke pemanggil API.

Di bagian berikut, kami menunjukkan cara mengatur respons gateway dengan menggunakan konsol API Gateway dan API Gateway REST API.

Menyiapkan respons gateway untuk menyesuaikan respons kesalahan

Jika API Gateway gagal memproses permintaan yang masuk, ia mengembalikan respons kesalahan ke klien tanpa meneruskan permintaan ke backend integrasi. Secara default, respons kesalahan berisi pesan kesalahan deskriptif singkat. Misalnya, jika Anda mencoba memanggil operasi pada sumber daya API yang tidak ditentukan, Anda menerima respons kesalahan dengan { "message": "Missing Authentication Token" } pesan tersebut. Jika Anda baru mengenal API Gateway, Anda mungkin merasa sulit untuk memahami apa yang sebenarnya salah.

Untuk beberapa respons kesalahan, API Gateway memungkinkan penyesuaian oleh pengembang API untuk mengembalikan respons dalam format yang berbeda. Missing Authentication Token Misalnya, Anda dapat menambahkan petunjuk ke payload respons asli dengan kemungkinan penyebabnya, seperti dalam contoh ini: `{"message": "Missing Authentication Token", "hint": "The HTTP method or resources may not be supported."}`

Saat API Anda memediasi antara pertukaran eksternal dan AWS Cloud, Anda menggunakan templat pemetaan VTL untuk permintaan integrasi atau respons integrasi untuk memetakan payload dari satu format ke format lainnya. Namun, template pemetaan VTL hanya berfungsi untuk permintaan yang valid dengan tanggapan yang berhasil.

Untuk permintaan yang tidak valid, API Gateway melewati integrasi sama sekali dan mengembalikan respons kesalahan. Anda harus menggunakan kustomisasi untuk membuat respons kesalahan dalam format yang sesuai dengan pertukaran. Di sini, kustomisasi diberikan dalam template pemetaan non-VTL yang hanya mendukung substitusi variabel sederhana.

Menggeneralisasi respons kesalahan yang dihasilkan API Gateway terhadap respons apa pun yang dihasilkan oleh API Gateway, kami menyebutnya sebagai respons gateway. Ini membedakan respons yang dihasilkan API Gateway dari respons integrasi. Template pemetaan respons gateway dapat mengakses nilai `$context` variabel dan nilai `$stageVariables` properti, serta parameter permintaan metode, dalam bentuk `method.request.param-position.param-name`

Untuk informasi lebih lanjut tentang `$context` variabel, lihat [Variabel konteks untuk transformasi data](#). Untuk informasi selengkapnya tentang `$stageVariables`, lihat [Variabel tahap](#). Untuk informasi selengkapnya tentang parameter permintaan metode, lihat [the section called “Variabel masukan”](#).

Topik

- [Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway](#)
- [Siapkan respons gateway menggunakan API Gateway REST API](#)
- [Siapkan kustomisasi respons gateway di OpenAPI](#)
- [Jenis respons gateway untuk API Gateway](#)

Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway

Contoh berikut menunjukkan cara menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway

Untuk menyesuaikan respons gateway menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Respons Gateway.
4. Pilih jenis respons, lalu pilih Edit. Dalam panduan ini, kami menggunakan token otentikasi Hilang sebagai contoh.
5. Anda dapat mengubah kode Status yang dihasilkan API Gateway untuk menampilkan kode status berbeda yang memenuhi persyaratan API Anda. Dalam contoh ini, kustomisasi mengubah kode status dari default (403) menjadi 404 karena pesan kesalahan ini terjadi ketika klien memanggil sumber daya yang tidak didukung atau tidak valid yang dapat dianggap sebagai tidak ditemukan.
6. Untuk mengembalikan header khusus, pilih Tambahkan header respons di bawah Header respons. Untuk tujuan ilustrasi, kami menambahkan header khusus berikut:

```
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
```

Dalam pemetaan header sebelumnya, nama domain statis ('a.b.c') dipetakan ke Allow-Control-Allow-Origin header untuk memungkinkan akses CORS ke API; header permintaan input x-amzn-RequestId dipetakan ke request-id dalam respons; variabel petId jalur permintaan yang masuk dipetakan ke request-path header dalam respons; dan parameter q kueri dari permintaan asli dipetakan ke header respons. request-query

7. Di bawah Template Response, simpan application/json untuk Jenis Konten dan masukkan template pemetaan tubuh berikut di editor badan Template:

```
{
  "message": "$context.error.messageString",
  "type": "$context.error.responseType",
  "statusCode": "'404'",
  "stage": "$context.stage",
  "resourcePath": "$context.resourcePath",
  "stageVariables.a": "$stageVariables.a"
}
```

Contoh ini menunjukkan cara memetakan `$context` dan `$stageVariables` properti ke properti badan respons gateway.

8. Pilih Simpan perubahan.
9. Menerapkan API ke tahap baru atau yang sudah ada.

Uji respons gateway Anda dengan memanggil perintah CURL berikut, dengan asumsi URL pemanggilan metode API yang sesuai adalah: `https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/{petId}`

```
curl -v -H 'x-amzn-RequestId:123344566' https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/5/type?q=1
```

Karena parameter string kueri tambahan `q=1` tidak kompatibel dengan API, kesalahan dikembalikan dari respons gateway yang ditentukan. Anda harus mendapatkan respons gateway yang mirip dengan yang berikut ini:

```
> 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-Mk93KgoxfzDlTUh3flmzA==

{
  "message": "Missing Authentication Token",
  "type": "MISSING_AUTHENTICATION_TOKEN",
```

```

    "statusCode": '404',
    "stage": custErr,
    "resourcePath": /pets/{petId},
    "stageVariables.a": a
  }

```

Contoh sebelumnya mengasumsikan bahwa backend API adalah [Pet Store](#) dan API memiliki variabel `stage`, didefinisikan. `a`

Siapkan respons gateway menggunakan API Gateway REST API

Sebelum menyesuaikan respons gateway menggunakan API Gateway REST API, Anda harus sudah membuat API dan telah memperoleh pengenalnya. Untuk mengambil pengenal API, Anda dapat mengikuti relasi tautan [restapi:gateway-response](#) dan memeriksa hasilnya.

Untuk menyesuaikan respons gateway menggunakan API Gateway REST API

1. Untuk menimpa seluruh [GatewayResponse](#) instance, panggil tindakan [gatewayresponse:put](#). Tentukan `responseType` yang diinginkan dalam parameter jalur URL, dan berikan dalam permintaan payload pemetaan `Status Code`, `ResponseParameters`, dan `ResponseTemplates`.
2. Untuk memperbarui bagian dari `GatewayResponse` instance, panggil tindakan [gatewayresponse:update](#). Tentukan parameter jalur URL yang diinginkan `responseType`, dan berikan dalam permintaan payload `GatewayResponse` properti individual yang Anda inginkan—misalnya, atau `responseParameters` pemetaan. `responseTemplates`

Siapkan kustomisasi respons gateway di OpenAPI

Anda dapat menggunakan `x-amazon-apigateway-gateway-responses` ekstensi di tingkat root API untuk menyesuaikan respons gateway di OpenAPI. Definisi OpenAPI berikut menunjukkan contoh untuk menyesuaikan [GatewayResponse](#) tipe. `MISSING_AUTHENTICATION_TOKEN`

```

"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}"
    }
  }
}

```

Dalam contoh ini, kustomisasi mengubah kode status dari default (403) menjadi 404. Ini juga menambah respons gateway empat parameter header dan satu template pemetaan tubuh untuk jenis `application/json` media.

Jenis respons gateway untuk API Gateway

API Gateway memaparkan respons gateway berikut untuk penyesuaian oleh pengembang API.

Jenis respons gateway	Kode status default	Deskripsi
ACCESS_DENIED	403	Respons gateway untuk kegagalan otorisasi—misalnya, saat akses ditolak oleh otorisasi khusus atau Amazon Cognito. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
API_CONFIGURATION_ERROR	500	Respons gateway untuk konfigurasi API yang tidak valid—termasuk saat alamat titik akhir yang tidak valid dikirimkan, saat decoding base64 gagal pada data biner saat dukungan biner diberlakukan, atau saat pemetaan respons integrasi tidak dapat cocok dengan templat apapun dan tidak ada templat default yang dikonfigurasi. Jika

Jenis respons gateway	Kode status default	Deskripsi
AUTHORIZER_CONFIGURATION_ERROR	500	tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX Respons gateway karena gagal terhubung ke otorisasi khusus atau Amazon Cognito. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
AUTHORIZER_FAILURE	500	Respons gateway saat otorisasi khusus atau Amazon Cognito gagal mengautentikasi pemanggil. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
BAD_REQUEST_PARAMETERS	400	Respons gateway ketika parameter permintaan tidak dapat divalidasi sesuai dengan validator permintaan yang diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
BAD_REQUEST_BODY	400	Respons gateway saat badan permintaan tidak dapat divalidasi sesuai dengan validator permintaan yang diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
DEFAULT_4XX	Null	Respons gateway default untuk jenis respons yang tidak ditentukan dengan kode status. 4XX Mengubah kode status respons gateway fallback ini mengubah kode status semua 4XX respons lain ke nilai baru. Menyetel ulang kode status ini ke null mengembalikan kode status dari semua 4XX respons lain ke nilai aslinya.  Note AWS WAF tanggapan khusus lebih diutamakan daripada respons gateway khusus.
DEFAULT_5XX	Null	Respons gateway default untuk jenis respons yang tidak ditentukan dengan kode status. 5XX Mengubah kode status respons gateway fallback ini mengubah kode status semua 5XX respons lain ke nilai baru. Menyetel ulang kode status ini ke null mengembalikan kode status dari semua 5XX respons lain ke nilai aslinya.

Jenis respons gateway	Kode status default	Deskripsi
EXPIRED_TOKEN	403	Respons gateway untuk kesalahan kedaluwarsa token AWS otentikasi. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
INTEGRATION_FAILURE	504	Respons gateway untuk kesalahan integrasi gagal. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
INTEGRATION_TIMEOUT	504	Respons gateway untuk kesalahan waktu integrasi habis. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
INVALID_API_KEY	403	Respons gateway untuk kunci API tidak valid yang dikirimkan untuk metode yang memerlukan kunci API. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
INVALID_SIGNATURE	403	Respons gateway untuk kesalahan AWS tanda tangan yang tidak valid. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
MISSING_AUTHENTICATION_TOKEN	403	Respons gateway untuk kesalahan token otentikasi yang hilang, termasuk kasus ketika klien mencoba memanggil metode atau sumber daya API yang tidak didukung. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
QUOTA_EXCEEDED	429	Respons gateway untuk kuota paket penggunaan melebihi kesalahan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
REQUEST_TOO_LARGE	413	Respons gateway untuk permintaan kesalahan terlalu besar. Jika jenis respons tidak ditentukan, respons ini default ke: HTTP content length exceeded 10485760 bytes

Jenis respons gateway	Kode status default	Deskripsi
RESOURCE_NOT_FOUND	404	Respons gateway ketika API Gateway tidak dapat menemukan sumber daya yang ditentukan setelah permintaan API melewati otentikasi dan otorisasi, kecuali untuk autentikasi dan otorisasi kunci API. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
THROTTLED	429	Respons gateway saat batas pelambatan tingkat rencana, metode, tahap, atau akun terlampaui. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
UNAUTHORIZED	401	Respons gateway saat otorisasi khusus atau Amazon Cognito gagal mengautentikasi pemanggil.
UNSUPPORTED_MEDIA_TYPE	415	Respons gateway ketika payload adalah tipe media yang tidak didukung, jika perilaku passthrough yang ketat diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
WAF_FILTERED	403	Respons gateway saat permintaan diblokir oleh AWS WAF. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

 **Note**
[AWS WAF tanggapan khusus](#) lebih diutamakan daripada respons gateway khusus.

CORS untuk REST APIs di API Gateway

[Cross-origin resource sharing \(CORS\)](#) adalah fitur keamanan browser yang membatasi permintaan HTTP lintas asal yang dimulai dari skrip yang berjalan di browser. Untuk informasi lebih lanjut, lihat [Apa itu CORS?](#).

Menentukan apakah akan mengaktifkan dukungan CORS

Permintaan HTTP lintas asal adalah permintaan yang dibuat untuk:

- Domain yang berbeda (misalnya, dari `example.com` ke `amazondomains.com`)
- Subdomain yang berbeda (misalnya, dari `example.com` ke `petstore.example.com`)
- Port yang berbeda (misalnya, dari `example.com` ke `example.com:10777`)
- Protokol yang berbeda (misalnya, dari `https://example.com` ke `http://example.com`)

Jika Anda tidak dapat mengakses API dan menerima pesan kesalahan yang berisi `Cross-Origin Request Blocked`, Anda mungkin perlu mengaktifkan CORS.

Permintaan HTTP cross-origin dapat dibagi menjadi dua jenis: permintaan sederhana dan permintaan non-sederhana.

Mengaktifkan CORS untuk permintaan sederhana

Permintaan HTTP sederhana jika semua kondisi berikut benar:

- Ini dikeluarkan terhadap sumber daya API yang hanya mengizinkan GET, HEAD, dan POST permintaan.
- Jika itu adalah permintaan POST metode, itu harus menyertakan Origin header.
- Jenis konten payload permintaan adalah text/plain, multipart/form-data, atau application/x-www-form-urlencoded.
- Permintaan tidak berisi header khusus.
- Persyaratan tambahan apa pun yang tercantum dalam [dokumentasi Mozilla CORS untuk permintaan sederhana](#).

Untuk permintaan POST metode lintas asal sederhana, respons dari sumber daya Anda harus menyertakan header Access-Control-Allow-Origin: '*' atau Access-Control-Allow-Origin: '*origin*'.

Semua permintaan HTTP lintas asal lainnya adalah permintaan non-sederhana.

Mengaktifkan CORS untuk permintaan yang tidak sederhana

Jika sumber daya API Anda menerima permintaan yang tidak sederhana, Anda harus mengaktifkan dukungan CORS tambahan tergantung pada jenis integrasi Anda.

Mengaktifkan CORS untuk integrasi non-proxy

Untuk integrasi ini, [protokol CORS](#) mengharuskan browser untuk mengirim permintaan preflight ke server dan menunggu persetujuan (atau permintaan kredensial) dari server sebelum mengirim permintaan yang sebenarnya. Anda harus mengonfigurasi API Anda untuk mengirim respons yang sesuai ke permintaan preflight.

Untuk membuat respons preflight:

1. Buat OPTIONS metode dengan integrasi tiruan.
2. Tambahkan header respons berikut ke respons metode 200:
 - Access-Control-Allow-Headers

- Access-Control-Allow-Methods
 - Access-Control-Allow-Origin
3. Atur perilaku passthrough integrasi ke NEVER. Dalam hal ini, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415. Untuk informasi selengkapnya, lihat [Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway](#).
 4. Masukkan nilai untuk header respons. Untuk mengizinkan semua asal, semua metode, dan header umum, gunakan nilai header berikut:
 - 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: '*'

Setelah membuat permintaan preflight, Anda harus mengembalikan Access-Control-Allow-Origin: '*origin*' header Access-Control-Allow-Origin: '*' or untuk semua metode yang mendukung CORS untuk setidaknya semua 200 tanggapan.

Mengaktifkan CORS untuk integrasi non-proxy menggunakan Konsol Manajemen AWS

Anda dapat menggunakan Konsol Manajemen AWS untuk mengaktifkan CORS. API Gateway membuat OPTIONS metode dan menambahkan Access-Control-Allow-Origin header ke respons integrasi metode yang ada. Ini tidak selalu berhasil, dan terkadang Anda perlu memodifikasi respons integrasi secara manual untuk mengembalikan Access-Control-Allow-Origin header untuk semua metode berkemampuan CORS untuk setidaknya semua 200 respons.

Jika Anda memiliki tipe media biner yang disetel ke */* API Anda, saat API Gateway membuat OPTIONS metode, ubah `contentHandling` ke `CONVERT_TO_TEXT`.

Perintah [update-integration](#) berikut mengubah ke `contentHandling` `CONVERT_TO_TEXT` untuk permintaan integrasi:

```
aws apigateway update-integration \  
  --rest-api-id abc123 \  
  --resource-id aaa111 \  
  --http-method OPTIONS \  
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

[update-integration-response](#) Perintah berikut mengubah `contentHandling` to `CONVERT_TO_TEXT` untuk respons integrasi:

```
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'
```

Mengaktifkan dukungan CORS untuk integrasi proxy

Untuk integrasi proxy Lambda atau integrasi proxy HTTP, backend Anda bertanggung jawab untuk mengembalikan `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods`, dan `Access-Control-Allow-Headers` header, karena integrasi proxy tidak mengembalikan respons integrasi.

Contoh berikut fungsi Lambda mengembalikan header CORS yang diperlukan:

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!')  
}
```

Topik

- [Aktifkan CORS pada sumber daya menggunakan konsol API Gateway](#)
- [Aktifkan CORS pada sumber daya menggunakan API impor API Gateway](#)
- [Uji CORS untuk API Gateway API](#)

Aktifkan CORS pada sumber daya menggunakan konsol API Gateway

Anda dapat menggunakan konsol API Gateway untuk mengaktifkan dukungan CORS untuk satu atau semua metode pada sumber daya REST API yang telah Anda buat. Setelah Anda mengaktifkan dukungan COR, atur perilaku passthrough integrasi ke. NEVER Dalam hal ini, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415. Untuk informasi selengkapnya, lihat [Perilaku permintaan metode untuk muatan tanpa memetakan template untuk REST API di API Gateway](#)

Important

Sumber daya dapat berisi sumber daya anak. Mengaktifkan dukungan CORS untuk sumber daya dan metodenya tidak secara rekursif mengaktifkannya untuk sumber daya anak dan metodenya.

Untuk mengaktifkan dukungan CORS pada sumber daya REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API.
3. Pilih sumber daya di bawah Sumber Daya.
4. Di bagian Rincian sumber daya, pilih Aktifkan CORS.

API Gateway > APIs > Resources - PetStore (abcd1234)

Resources

API actions ▼ Deploy API

Create resource

- /
 - GET
 - /pets
 - GET
 - OPTIONS
 - POST
 - /petId
 - GET
 - OPTIONS

Resource details

Path / Resource ID efg456

Update documentation **Enable CORS**

Methods (1)

Delete Create method

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

5. Di kotak Aktifkan CORS, lakukan hal berikut:

- (Opsional) Jika Anda membuat respons gateway khusus dan ingin mengaktifkan dukungan CORS untuk respons, pilih respons gateway.
- Pilih setiap metode untuk mengaktifkan dukungan CORS. `OPTION` Metode ini harus mengaktifkan CORS.

Jika Anda mengaktifkan dukungan CORS untuk suatu ANY metode, CORS diaktifkan untuk semua metode.

- Di bidang input Access-Control-Allow-Headers, masukkan string statis dari daftar header yang dipisahkan koma yang harus dikirimkan klien dalam permintaan sumber daya yang sebenarnya. Gunakan daftar header yang disediakan konsol 'Content-Type,X-Amz-Date,Authorization,X-API-Key,X-Amz-Security-Token' atau tentukan header Anda sendiri.
- Gunakan nilai yang disediakan konsol ' * ' sebagai nilai header Access-Control-Allow-Origin untuk mengizinkan permintaan akses dari semua asal, atau tentukan asal yang akan diizinkan mengakses sumber daya.
- Pilih Simpan.

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

Saat menerapkan instruksi di atas ke ANY metode dalam integrasi proxy, header CORS apa pun yang berlaku tidak akan disetel. Sebagai gantinya, backend Anda harus mengembalikan header CORS yang berlaku, seperti. Access-Control-Allow-Origin

Setelah CORS diaktifkan pada GET metode, OPTIONS metode ditambahkan ke sumber daya, jika belum ada. 200Respons OPTIONS metode ini secara otomatis dikonfigurasi untuk mengembalikan tiga Access-Control-Allow-* header untuk memenuhi jabat tangan preflight. Selain itu, metode aktual (GET) juga dikonfigurasi secara default untuk mengembalikan Access-Control-Allow-Origin header dalam respons 200 juga. Untuk jenis tanggapan lain, Anda perlu mengonfigurasinya secara manual untuk mengembalikan Access-Control-Allow-Origin' header dengan '*' atau asal tertentu, jika Anda tidak ingin mengembalikan Cross-origin access kesalahan.

Setelah Anda mengaktifkan dukungan CORS pada sumber daya Anda, Anda harus menerapkan atau menerapkan ulang API agar pengaturan baru diterapkan. Untuk informasi selengkapnya, lihat [the section called “Buat deployment”](#).

Note

Jika Anda tidak dapat mengaktifkan dukungan CORS pada sumber daya Anda setelah mengikuti prosedur, kami sarankan Anda membandingkan konfigurasi CORS Anda dengan sumber daya API /pets contoh. Untuk mempelajari cara membuat contoh API, lihat [the section called “Tutorial: Buat REST API dengan mengimpor contoh”](#).

Aktifkan CORS pada sumber daya menggunakan API impor API Gateway

Jika Anda menggunakan [API Gateway Import API](#), Anda dapat mengatur dukungan CORS menggunakan file OpenAPI. Anda harus terlebih dahulu menentukan OPTIONS metode dalam sumber daya Anda yang mengembalikan header yang diperlukan.

Note

Browser web mengharapkan Access-Control-Allow-Headers, dan Access-Control-Allow-Origin header akan diatur di setiap metode API yang menerima permintaan CORS. Selain itu, beberapa browser pertama-tama membuat permintaan HTTP ke OPTIONS metode di sumber daya yang sama, dan kemudian berharap untuk menerima header yang sama.

OptionsMetode contoh

Contoh berikut menciptakan OPTIONS metode untuk integrasi tiruan.

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"

```

Setelah Anda mengonfigurasi OPTIONS metode untuk sumber daya Anda, Anda dapat menambahkan header yang diperlukan ke metode lain di sumber daya yang sama yang perlu menerima permintaan CORS.

1. Deklarasikan Access-Control-Allow-Origin dan Header ke tipe respons.

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. Di `x-amazon-apigateway-integration` tag, atur pemetaan untuk header tersebut ke nilai statis Anda:

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 : "'*'"
```

Contoh API

Contoh berikut membuat API lengkap dengan OPTIONS metode dan GET metode dengan HTTP integrasi.

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"

```

Uji CORS untuk API Gateway API

Anda dapat menguji konfigurasi CORS API Anda dengan menjalankan API Anda, dan memeriksa header CORS dalam respons. `curl` Perintah berikut mengirimkan permintaan OPTIONS ke API yang diterapkan.

```
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

```

Access-Control-Allow-MethodsHeader Access-Control-Allow-OriginAccess-Control-Allow-Headers,, dan dalam respons menunjukkan bahwa API mendukung CORS. Lihat informasi yang lebih lengkap di [CORS untuk REST APIs di API Gateway](#).

Jenis media biner untuk REST APIs di API Gateway

Di API Gateway, permintaan dan respons API memiliki muatan teks atau biner. Payload teks adalah string JSON UTF-8 -encoded. Payload biner adalah apa pun selain payload teks. Payload biner

dapat berupa, misalnya, file JPEG, file, atau GZip file XHTML. Konfigurasi API yang diperlukan untuk mendukung media biner bergantung pada apakah API Anda menggunakan integrasi proxy atau non-proxy.

Jika Anda menggunakan integrasi proxy dengan streaming respons payload, Anda tidak perlu mengonfigurasi jenis media biner Anda. Untuk informasi selengkapnya, lihat [the section called “Mengalirkan respons integrasi untuk integrasi proksi Anda”](#).

AWS Lambda integrasi proxy

Untuk menangani payload biner untuk integrasi AWS Lambda proxy, Anda harus base64-menyandikan respons fungsi Anda. Anda juga harus mengonfigurasi [binaryMediaTypes](#) untuk API Anda. `binaryMediaTypes` konfigurasi API Anda adalah daftar tipe konten yang API Anda perlakukan sebagai data biner. Contoh jenis media biner termasuk `image/png` atau `application/octet-stream`. Anda dapat menggunakan karakter wildcard (*) untuk mencakup beberapa jenis media.

API Gateway menggunakan Accept header pertama dari klien untuk menentukan apakah respons harus mengembalikan media biner. Untuk mengembalikan media biner saat Anda tidak dapat mengontrol urutan nilai Accept header, seperti permintaan dari browser, setel tipe media biner API Anda */*.

Untuk kode sampel, lihat [the section called “Kembalikan media biner dari integrasi proxy Lambda di API Gateway”](#).

Jika Anda menggunakan integrasi proxy Lambda dengan streaming respons payload, Anda tidak perlu mengonfigurasi jenis media biner Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan integrasi proksi Lambda dengan aliran respons payload”](#).

Integrasi non-proxy

Untuk menangani muatan biner untuk integrasi non-proxy, Anda menambahkan jenis media ke [binaryMediaTypes](#) daftar sumber daya. RestApi `binaryMediaTypes` konfigurasi API Anda adalah daftar tipe konten yang API Anda perlakukan sebagai data biner. [Atau, Anda dapat mengatur properti ContentHandling pada Integrasi dan sumber daya. IntegrationResponse](#) `contentHandling` nilainya bisa `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT`, atau tidak terdefinisi.

 Note

Integrasi untuk MOCK atau pribadi, pengaturan `contentHandling` properti tidak didukung di. Konsol Manajemen AWS Anda harus menggunakan AWS CLI, CloudFormation, atau SDK untuk mengatur `contentHandling` properti.

Bergantung pada `contentHandling` nilainya, dan apakah header respons atau `Content-Type` header permintaan yang masuk cocok dengan entri dalam `binaryMediaTypes` daftar, API Gateway dapat menyandikan byte biner mentah sebagai string yang disandikan base64, memecahkan kode string yang dikodekan base64 kembali ke byte mentahnya, atau meneruskan isi tanpa modifikasi. `Accept`

Anda harus mengonfigurasi API sebagai berikut untuk mendukung payload biner untuk API Anda di API Gateway:

- Tambahkan jenis media biner yang diinginkan ke `binaryMediaTypes` daftar pada [RestApi](#) sumber daya. Jika properti ini dan properti tidak ditentukan, muatan ditangani sebagai string JSON yang dikodekan UTF-8. `contentHandling`
- Alamat `contentHandling` properti sumber daya [Integrasi](#).
 - Agar payload permintaan dikonversi dari string yang dikodekan base64 ke gumpalan binernya, setel properti ke. `CONVERT_TO_BINARY`
 - Agar payload permintaan dikonversi dari gumpalan biner ke string yang dikodekan base64, setel properti ke. `CONVERT_TO_TEXT`
 - Untuk meneruskan muatan tanpa modifikasi, biarkan properti tidak terdefinisi. Untuk meneruskan payload biner tanpa modifikasi, Anda juga harus memastikan bahwa entri tersebut `Content-Type` cocok dengan salah satu `binaryMediaTypes` entri, dan [perilaku passthrough](#) diaktifkan untuk API.
- Atur `contentHandling` properti sumber [IntegrationResponse](#) daya. `contentHandlingProperty`, `Accept` header dalam permintaan klien, dan `binaryMediaTypes` gabungan API Anda menentukan cara API Gateway menangani konversi tipe konten. Lihat perinciannya di [the section called “Konversi jenis konten di API Gateway”](#).

⚠ Important

Jika permintaan berisi beberapa jenis media di Accept header, API Gateway hanya menghormati jenis Accept media pertama. Jika Anda tidak dapat mengontrol urutan jenis Accept media dan jenis media konten biner Anda bukan yang pertama dalam daftar, tambahkan jenis Accept media pertama dalam `binaryMediaTypes` daftar API Anda. API Gateway menangani semua jenis konten dalam daftar ini sebagai biner.

Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir menerima file JPEG sebagai biner.

Untuk informasi mendetail tentang cara API Gateway menangani muatan teks dan biner, lihat [Konversi jenis konten di API Gateway](#).

Konversi jenis konten di API Gateway

Kombinasi API Anda `binaryMediaTypes`, header dalam permintaan klien, dan `contentHandling` properti integrasi menentukan cara API Gateway menyandikan payload.

Tabel berikut menunjukkan cara API Gateway mengonversi payload permintaan untuk konfigurasi spesifik Content-Type header permintaan, `binaryMediaTypes` daftar RestApi sumber daya, dan nilai `contentHandling` properti sumber daya Integrasi.

Metode permintaan payload	Permintaan Content-Type header	binaryMediaTypes	contentHandling	Muatan permintaan integrasi
Data teks	Tipe data apa pun	Tidak terdefinisi	Tidak terdefinisi	UTF8-string yang dikodekan
Data teks	Tipe data apa pun	Tidak terdefinisi	CONVERT_TO_BINARY	Gumpalan biner yang didekode Base64
Data teks	Tipe data apa pun	Tidak terdefinisi	CONVERT_TO_TEXT	UTF8-string yang dikodekan

Metode permintaan payload	Permintaan Content-Type header	binaryMediaTypes	contentHandling	Muatan permintaan integrasi
Data teks	Tipe data teks	Atur dengan jenis media yang cocok	Tidak terdefinisi	Data teks
Data teks	Tipe data teks	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Gumpalan biner yang didekode Base64
Data teks	Tipe data teks	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	Data teks
Data biner	Tipe data biner	Atur dengan jenis media yang cocok	Tidak terdefinisi	Data biner
Data biner	Tipe data biner	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Data biner
Data biner	Tipe data biner	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	String yang dikodekan Base64

Tabel berikut menunjukkan cara API Gateway mengonversi payload respons untuk konfigurasi spesifik Accept header permintaan, `binaryMediaTypes` daftar [RestApi](#) sumber daya, dan nilai `contentHandling` properti sumber daya. [IntegrationResponse](#)

Important

Jika permintaan berisi beberapa jenis media di Accept header, API Gateway hanya menghormati jenis Accept media pertama. Jika Anda tidak dapat mengontrol urutan jenis Accept media dan jenis media konten biner Anda bukan yang pertama dalam daftar,

tambahkan jenis `Accept` media pertama dalam `binaryMediaTypes` daftar API Anda. API Gateway menangani semua jenis konten dalam daftar ini sebagai biner.

Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir menerima file JPEG sebagai biner.

Payload respons integrasi	Permintaan Accept header	binaryMediaTypes	contentHandling	Metode respon payload
Teks atau data biner	Jenis teks	Tidak terdefinisi	Tidak terdefinisi	UTF8-string yang dikodekan
Teks atau data biner	Jenis teks	Tidak terdefinisi	CONVERT_TO_BINARY	Gumpalan yang didekode Base64
Teks atau data biner	Jenis teks	Tidak terdefinisi	CONVERT_TO_TEXT	UTF8-string yang dikodekan
Data teks	Jenis teks	Atur dengan jenis media yang cocok	Tidak terdefinisi	Data teks
Data teks	Jenis teks	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Gumpalan yang didekode Base64
Data teks	Jenis teks	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	UTF8-string yang dikodekan
Data teks	Tipe biner	Atur dengan jenis media yang cocok	Tidak terdefinisi	Gumpalan yang didekode Base64

Payload respons integrasi	Permintaan Accept header	binaryMediaTypes	contentHandling	Metode respon payload
Data teks	Tipe biner	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Gumpalan yang didekode Base64
Data teks	Tipe biner	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	UTF8-string yang dikodekan
Data biner	Jenis teks	Atur dengan jenis media yang cocok	Tidak terdefinisi	String yang dikodekan Base64
Data biner	Jenis teks	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Data biner
Data biner	Jenis teks	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	String yang dikodekan Base64
Data biner	Tipe biner	Atur dengan jenis media yang cocok	Tidak terdefinisi	Data biner
Data biner	Tipe biner	Atur dengan jenis media yang cocok	CONVERT_TO_BINARY	Data biner
Data biner	Tipe biner	Atur dengan jenis media yang cocok	CONVERT_TO_TEXT	String yang dikodekan Base64

Saat mengonversi payload teks menjadi gumpalan biner, API Gateway mengasumsikan bahwa data teks adalah string yang dikodekan base64 dan mengeluarkan data biner sebagai gumpalan yang diterjemahkan base64. Jika konversi gagal, ia mengembalikan 500 respons, yang menunjukkan

kesalahan konfigurasi API. Anda tidak menyediakan template pemetaan untuk konversi semacam itu, meskipun Anda harus mengaktifkan [perilaku passthrough](#) di API.

Saat mengonversi payload biner menjadi string teks, API Gateway selalu menerapkan pengkodean base64 pada data biner. Anda dapat menentukan templat pemetaan untuk muatan semacam itu, tetapi hanya dapat mengakses string yang dikodekan base64 dalam templat pemetaan `$input.body`, seperti yang ditunjukkan dalam kutipan contoh templat pemetaan berikut.

```
{
  "data": "$input.body"
}
```

Agar payload biner melewati tanpa modifikasi, Anda harus mengaktifkan [perilaku passthrough](#) di API.

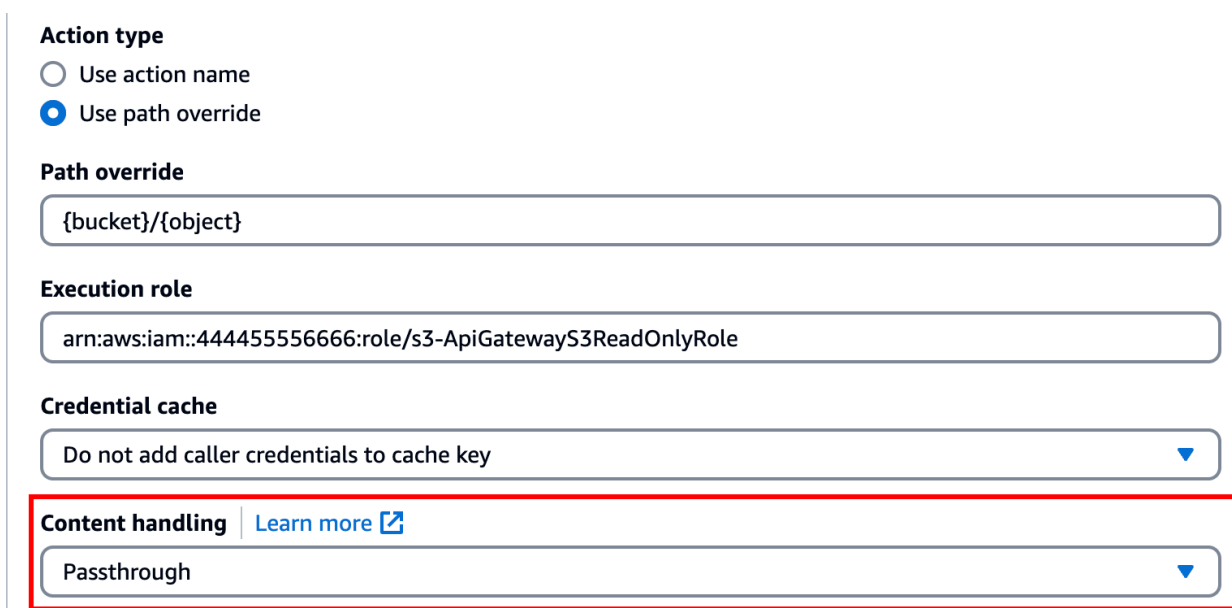
Mengaktifkan dukungan biner menggunakan konsol API Gateway

Bagian ini menjelaskan cara mengaktifkan dukungan biner menggunakan konsol API Gateway. Sebagai contoh, kami menggunakan API yang terintegrasi dengan Amazon S3. Kami fokus pada tugas untuk mengatur jenis media yang didukung dan untuk menentukan bagaimana payload harus ditangani. Untuk informasi terperinci tentang cara membuat API yang terintegrasi dengan Amazon S3, lihat. [Tutorial: Buat REST API sebagai proxy Amazon S3](#)

Untuk mengaktifkan dukungan biner dengan menggunakan konsol API Gateway

1. Tetapkan tipe media biner untuk API:
 - a. Buat API baru atau pilih API yang sudah ada. Untuk contoh ini, kami memberi nama `APIFileMan`.
 - b. Di bawah API yang dipilih di panel navigasi utama, pilih pengaturan API.
 - c. Di panel pengaturan API, pilih Kelola jenis media di bagian Jenis Media Biner.
 - d. Pilih Tambahkan tipe media biner.
 - e. Masukkan jenis media yang diperlukan, misalnya `image/png`, di bidang teks input. Jika perlu, ulangi langkah ini untuk menambahkan lebih banyak jenis media. Untuk mendukung semua jenis media biner, tentukan `*/*`.
 - f. Pilih Simpan perubahan.
2. Mengatur cara payload pesan ditangani untuk metode API:

- a. Buat yang baru atau pilih sumber daya yang ada di API. Untuk contoh ini, kami menggunakan `/ {folder} / {item}` sumber daya.
- b. Buat yang baru atau pilih metode yang ada pada sumber daya. Sebagai contoh, kami menggunakan `GET / {folder} / {item}` metode yang terintegrasi dengan `Object GET` tindakan di Amazon S3.
- c. Untuk penanganan Konten, pilih opsi.



The screenshot shows the configuration interface for an API method in Amazon API Gateway. It includes several sections: 'Action type' with radio buttons for 'Use action name' and 'Use path override' (selected); 'Path override' with a text input field containing '{bucket}/{object}'; 'Execution role' with a text input field containing 'arn:aws:iam::444455556666:role/s3-ApiGatewayS3ReadOnlyRole'; 'Credential cache' with a dropdown menu showing 'Do not add caller credentials to cache key'; and 'Content handling' with a dropdown menu showing 'Passthrough'. The 'Content handling' section is highlighted with a red border. A 'Learn more' link is visible next to the 'Content handling' label.

Pilih Passthrough jika Anda tidak ingin mengonversi isi ketika klien dan backend menerima format biner yang sama. Pilih Konversi ke teks untuk mengonversi badan biner menjadi string yang dikodekan base64 ketika, misalnya, backend mengharuskan payload permintaan biner diteruskan sebagai properti JSON. Dan pilih Konversi ke biner ketika klien mengirimkan string yang dikodekan base64 dan backend memerlukan format biner asli, atau ketika titik akhir mengembalikan string yang dikodekan base64 dan klien hanya menerima output biner.

- d. Untuk Passthrough badan Permintaan, pilih Bila tidak ada templat yang ditentukan (disarankan) untuk mengaktifkan perilaku passthrough pada badan permintaan.

Anda juga bisa memilih Never. Ini berarti bahwa API akan menolak data dengan tipe konten yang tidak memiliki template pemetaan.

- e. Pertahankan Accept header permintaan masuk dalam permintaan integrasi. Anda harus melakukan ini jika Anda telah mengatur passthrough dan `contentHandling` ingin mengganti pengaturan itu saat runtime.

HTTP headers (2)			< 1 >
Name	Mapped from	Caching	
Accept	method.request.header.Accept	Inactive	
Content-Type	method.request.header.Content-Type	Inactive	

- f. Untuk konversi ke teks, tentukan template pemetaan untuk menempatkan data biner yang dikodekan base64 ke dalam format yang diperlukan.

Contoh template pemetaan untuk dikonversi ke teks adalah sebagai berikut:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

Format template pemetaan ini tergantung pada persyaratan titik akhir input.

- g. Pilih Simpan.

Mengaktifkan dukungan biner menggunakan API Gateway REST API

Tugas berikut menunjukkan cara mengaktifkan dukungan biner menggunakan panggilan API Gateway REST API.

Topik

- [Menambahkan dan memperbarui tipe media biner yang didukung ke API](#)
- [Konfigurasi konversi payload permintaan](#)
- [Konfigurasi konversi payload respons](#)
- [Ubah data biner menjadi data teks](#)
- [Mengkonversi data teks ke payload biner](#)
- [Melewati muatan biner](#)

Menambahkan dan memperbarui tipe media biner yang didukung ke API

Untuk mengaktifkan API Gateway untuk mendukung tipe media biner baru, Anda harus menambahkan jenis media biner ke `binaryMediaTypes` daftar `RestApi` sumber daya. Misalnya, agar API Gateway menangani gambar JPEG, kirimkan PATCH permintaan ke `RestApi` sumber daya:

```

PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
]
}

```

Spesifikasi tipe MIME `image/jpeg` yang merupakan bagian dari nilai path properti diloloskan sebagai `image~1jpeg`

Untuk memperbarui jenis media biner yang didukung, ganti atau hapus jenis media dari `binaryMediaTypes` daftar RestApi sumber daya. Misalnya, untuk mengubah dukungan biner dari file JPEG ke byte mentah, kirimkan PATCH permintaan ke RestApi sumber daya, sebagai berikut:

```

PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
]
}

```

Konfigurasi konversi payload permintaan

Jika titik akhir membutuhkan input biner, atur `contentHandling` properti Integration sumber daya ke `CONVERT_TO_BINARY`. Untuk melakukannya, kirimkan PATCH permintaan, sebagai berikut:

```

PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",

```

```

    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Konfigurasi konversi payload respons

Jika klien menerima hasilnya sebagai gumpalan biner alih-alih muatan yang dikodekan base64 yang dikembalikan dari titik akhir, setel properti sumber daya ke. `contentHandling` `IntegrationResponse` `CONVERT_TO_BINARY` Untuk melakukan ini, kirimkan PATCH permintaan, sebagai berikut:

```

PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Ubah data biner menjadi data teks

Untuk mengirim data biner sebagai properti JSON dari input ke AWS Lambda atau Kinesis melalui API Gateway, lakukan hal berikut:

1. Aktifkan dukungan payload biner API dengan menambahkan jenis media biner baru `application/octet-stream` ke `binaryMediaTypes` daftar API.

```

PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }]
}
```

2. Tetapkan `CONVERT_TO_TEXT` pada `contentHandling` properti `Integration` sumber daya dan sediakan template pemetaan untuk menetapkan string data biner yang dikodekan base64 ke

properti JSON. Dalam contoh berikut, properti JSON adalah `body` dan `$input.body` memegang string yang dikodekan base64.

```
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}\"}"
    }
  ]
}
```

Mengkonversi data teks ke payload biner

Misalkan fungsi Lambda mengembalikan file gambar sebagai string yang dikodekan base64. Untuk meneruskan output biner ini ke klien melalui API Gateway, lakukan hal berikut:

1. Perbarui `binaryMediaTypes` daftar API dengan menambahkan jenis media `binerapplication/octet-stream`, jika belum ada dalam daftar.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Setel `contentHandling` properti pada `Integration` sumber daya ke `CONVERT_TO_BINARY`. Jangan mendefinisikan template pemetaan. Jika Anda tidak mendefinisikan template pemetaan,

API Gateway akan memanggil template passthrough untuk mengembalikan blob biner yang diterjemahkan base64 sebagai file gambar ke klien.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Melewati muatan biner

Untuk menyimpan gambar di bucket Amazon S3 menggunakan API Gateway, lakukan hal berikut:

1. Perbarui `binaryMediaTypes` daftar API dengan menambahkan jenis media `binerapplication/octet-stream`, jika belum ada dalam daftar.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
]
```

2. Pada `contentHandling` properti sumber `Integration` daya, atur `CONVERT_TO_BINARY`. Tetapkan `WHEN_NO_MATCH` sebagai nilai `passthroughBehavior` properti tanpa mendefinisikan template pemetaan. Hal ini memungkinkan API Gateway untuk memanggil template passthrough.

```
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"  
  }  
]  
}
```

Impor dan ekspor pengkodean konten untuk API Gateway

Untuk mengimpor `binaryMediaTypes` daftar di a [RestApi](#), gunakan ekstensi API Gateway berikut ke file definisi OpenAPI API. Ekstensi ini juga digunakan untuk mengekspor pengaturan API.

- [x-amazon-apigateway-binaryproperty -media-tipe](#)

Untuk mengimpor dan mengekspor nilai `contentHandling` properti pada `IntegrationResponse` sumber daya `Integration` atau, gunakan ekstensi API Gateway berikut ke definisi OpenAPI:

- [x-amazon-apigateway-integration objek](#)
- [x-amazon-apigateway-integration.response objek](#)

Kembalikan media biner dari integrasi proxy Lambda di API Gateway

Untuk mengembalikan media biner dari [integrasi AWS Lambda proxy](#), base64 menyandikan respons dari fungsi Lambda Anda. Anda juga harus [mengonfigurasi tipe media biner API Anda](#). Saat Anda mengonfigurasi tipe media biner API Anda, API Anda memperlakukan tipe konten tersebut sebagai data biner. Batas ukuran muatan adalah 10 MB.

Note

Untuk menggunakan browser web untuk menjalankan API dengan contoh integrasi ini, setel tipe media biner API Anda ke `*/*`. API Gateway menggunakan Accept header pertama dari klien untuk menentukan apakah respons harus mengembalikan media biner. Untuk

mengembalikan media biner saat Anda tidak dapat mengontrol urutan nilai Accept header, seperti permintaan dari browser, setel tipe media biner API Anda ke `*/*` (untuk semua jenis konten).

Contoh berikut fungsi Lambda dapat mengembalikan gambar biner dari Amazon S3 atau teks ke klien. Respons fungsi mencakup Content-Type header untuk menunjukkan kepada klien jenis data yang dikembalikan. Fungsi secara kondisional menetapkan `isBase64Encoded` properti dalam responsnya, tergantung pada jenis data yang dikembalikan.

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>",
        }
```

Untuk mempelajari lebih lanjut tentang jenis media biner, lihat [Jenis media biner untuk REST APIs di API Gateway](#).

Akses file biner di Amazon S3 melalui API Gateway API

Contoh berikut menunjukkan file OpenAPI yang digunakan untuk mengakses gambar di Amazon S3, cara mengunduh gambar dari Amazon S3, dan cara mengunggah gambar ke Amazon S3.

Topik

- [File OpenAPI dari API sampel untuk mengakses gambar di Amazon S3](#)
- [Unduh gambar dari Amazon S3](#)
- [Unggah gambar ke Amazon S3](#)

File OpenAPI dari API sampel untuk mengakses gambar di Amazon S3

File OpenAPI berikut menunjukkan contoh API yang menggambarkan mengunduh file gambar dari Amazon S3 dan mengunggah file gambar ke Amazon S3. API ini mengekspos GET `/s3?key={file-name}` dan PUT `/s3?key={file-name}` metode untuk mengunduh dan mengunggah file gambar tertentu. GETMetode mengembalikan file gambar sebagai string yang dikodekan base64 sebagai bagian dari output JSON, mengikuti template pemetaan yang disediakan, dalam respons 200 OK. PUTMetode ini mengambil gumpalan biner mentah sebagai input dan mengembalikan respons 200 OK dengan muatan kosong.

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"
    }
}

```

```

    }
  }
}

```

Unduh gambar dari Amazon S3

Untuk mengunduh file gambar (`image.jpg`) sebagai gumpalan biner dari Amazon S3:

```

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

```

Respons yang berhasil terlihat seperti ini:

```

200 OK HTTP/1.1

[raw bytes]

```

Byte mentah dikembalikan karena `Accept` header disetel ke jenis media biner `application/octet-stream` dan dukungan biner diaktifkan untuk API.

Atau, untuk mengunduh file gambar (`image.jpg`) sebagai string yang disandikan base64 (diformat sebagai properti JSON) dari Amazon S3, tambahkan template respons ke respons integrasi 200, seperti yang ditunjukkan pada blok definisi OpenAPI berwajah tebal berikut:

```

"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}"
      }
    }
  },
}

```

Permintaan untuk mengunduh file gambar terlihat seperti berikut:

```
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
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Unggah gambar ke Amazon S3

Untuk mengunggah file gambar (image.jpg) sebagai gumpalan biner ke Amazon S3:

```
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]
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1
```

Untuk mengunggah file gambar (image.jpg) sebagai string yang dikodekan base64 ke Amazon S3:

```
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=
```

Muatan input harus berupa string yang dikodekan base64 karena nilai Content-Type header diatur ke application/json Respons yang berhasil terlihat seperti berikut:


```
200 OK HTTP/1.1
```

Akses file biner di Lambda menggunakan API Gateway API

Contoh OpenAPI berikut menunjukkan cara mengakses file biner AWS Lambda melalui API Gateway API. API ini mengekspos GET `/lambda?key={file-name}` dan PUT `/lambda?key={file-name}` metode untuk mengunduh dan mengunggah file gambar tertentu. GETMetode mengembalikan file gambar sebagai string yang dikodekan base64 sebagai bagian dari output JSON, mengikuti template pemetaan yang disediakan, dalam respons 200 OK. PUTMetode ini mengambil gumpalan biner mentah sebagai input dan mengembalikan respons 200 OK dengan muatan kosong.

Anda membuat fungsi Lambda yang dipanggil API Anda, dan itu harus mengembalikan string yang dikodekan base64 dengan header. Content-Type application/json

Topik

- [File OpenAPI dari API sampel untuk mengakses gambar di Lambda](#)
- [Unduh gambar dari Lambda](#)
- [Unggah gambar ke Lambda](#)

File OpenAPI dari API sampel untuk mengakses gambar di Lambda

File OpenAPI berikut menunjukkan contoh API yang menggambarkan mengunduh file gambar dari Lambda dan mengunggah file gambar ke Lambda.

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": {
                        "$ref": "#/components/schemas/Empty"
                    }
                }
            }
        },
        "500": {
            "description": "500 response"
        }
    },
    "x-amazon-apigateway-integration": {
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
        "type": "AWS",
        "credentials": "arn:aws:iam::123456789012:role/Lambda",
        "httpMethod": "POST",
        "requestTemplates": {
            "application/json": "{\n    \"imageKey\":\n    \"${input.params('key')}\",\n    \"image\": \"${input.body}\"}"
        },
        "responses": {
            "default": {
                "statusCode": "500"
            },
            "2\\d{2}": {
                "statusCode": "200",
                "responseTemplates": {
                    "application/json": "{\n    \"image\": \"${input.body}\"}"
                }
            }
        }
    },
    "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\": \"${input.params('key')}\",\n  \"image\": \"${input.body}\"}"
      },
      "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\": \"${input.params('key')}\",\n  \"image\": \"${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"
    }
}
}
}

```

Unduh gambar dari Lambda

Untuk mengunduh file gambar (image.jpg) sebagai gumpalan biner dari Lambda:

```

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

```

Respons yang berhasil terlihat seperti berikut:

```

200 OK HTTP/1.1

[raw bytes]

```

Untuk mengunduh file gambar (image.jpg) sebagai string yang dikodekan base64 (diformat sebagai properti JSON) dari Lambda:

```

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

```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Unggah gambar ke Lambda

Untuk mengunggah file gambar (`image.jpg`) sebagai gumpalan biner ke Lambda:

```
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]
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK
```

Untuk mengunggah file gambar (`image.jpg`) sebagai string yang dikodekan base64 ke Lambda:

```
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=
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK
```

Panggil REST APIs di API Gateway

Untuk memanggil API yang diterapkan, klien mengirimkan permintaan ke URL untuk layanan komponen API Gateway untuk eksekusi API, yang dikenal sebagai `execute-api`.

URL dasar untuk REST APIs adalah dalam format berikut:

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

di *api-id* mana pengenalan API, *region* adalah AWS Wilayah, dan *stage* merupakan nama panggung penerapan API.

Important

Sebelum Anda dapat menjalankan API, Anda harus menerapkannya di API Gateway. Untuk petunjuk tentang penerapan API, lihat [Menerapkan REST APIs di API Gateway](#).

Topik

- [Mendapatkan URL pemanggilan API](#)
- [Memanggil API](#)
- [Menggunakan konsol API Gateway untuk menguji metode REST API](#)
- [Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API](#)
- [Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API](#)
- [Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API](#)
- [Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API](#)
- [Menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift](#)

Mendapatkan URL pemanggilan API

Anda dapat menggunakan konsol, definisi OpenAPI AWS CLI, atau yang diekspor untuk mendapatkan URL pemanggilan API.

Mendapatkan URL pemanggilan API menggunakan konsol

Prosedur berikut menunjukkan cara mendapatkan URL pemanggilan API di konsol REST API.

Untuk mendapatkan URL pemanggilan API menggunakan konsol REST API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pilih API yang diterapkan.
3. Dari panel navigasi utama, pilih Stage.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

URL ini untuk sumber daya root API Anda.

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. Untuk mendapatkan URL pemanggilan API untuk sumber daya lain di API Anda, perluas tahapan di bawah panel navigasi sekunder, lalu pilih metode.
6. Pilih ikon salin untuk menyalin URL pemanggilan tingkat sumber daya API Anda.

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

Mendapatkan URL pemanggilan API menggunakan AWS CLI

Prosedur berikut menunjukkan cara mendapatkan URL pemanggilan API menggunakan. AWS CLI

Untuk mendapatkan URL pemanggilan API menggunakan AWS CLI

1. Gunakan perintah berikut untuk mendapatkan `rest-api-id`. Perintah ini mengembalikan semua `rest-api-id` nilai di Wilayah Anda. Untuk informasi selengkapnya, lihat [get-rest-apis](#).

```
aws apigateway get-rest-apis
```

2. Ganti contoh `rest-api-id` dengan `Andarest-api-id`, ganti contoh `{stage-name}` dengan `Anda{stage-name}`, dan ganti `{region}`, dengan Wilayah Anda.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Memperoleh URL pemanggilan API menggunakan file definisi OpenAPI yang diekspor dari API

Anda juga dapat membuat URL root dengan menggabungkan host dan basePath bidang file definisi OpenAPI yang diekspor dari API. Untuk petunjuk tentang cara mengekspor API Anda, lihat [the section called “Ekspor REST API”](#).

Memanggil API

[Anda dapat memanggil API yang digunakan menggunakan browser, curl, atau aplikasi lain, seperti Postman.](#)

Selain itu, Anda dapat menggunakan konsol API Gateway untuk menguji panggilan API. Pengujian menggunakan TestInvoke fitur API Gateway, yang memungkinkan pengujian API sebelum API diterapkan. Untuk informasi selengkapnya, lihat [the section called “Gunakan konsol untuk menguji metode REST API”](#).

Note

Nilai parameter string kueri dalam URL pemanggilan tidak dapat berisi. %%

Memanggil API menggunakan browser web

Jika API Anda mengizinkan akses anonim, Anda dapat menggunakan browser web apa pun untuk menjalankan metode apa pun GET. Masukkan URL pemanggilan lengkap di bilah alamat browser.

Untuk metode lain atau panggilan yang diperlukan otentikasi, Anda harus menentukan payload atau menandatangani permintaan. Anda dapat menangani ini dalam skrip di belakang halaman HTML atau dalam aplikasi klien menggunakan salah satu file AWS SDKs.

Memanggil API menggunakan curl

Anda dapat menggunakan alat seperti [curl](#) di terminal Anda untuk memanggil API Anda. Contoh perintah curl berikut memanggil metode GET pada getUsers sumber daya prod tahap API.

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"
```

Menggunakan konsol API Gateway untuk menguji metode REST API

Gunakan konsol API Gateway untuk menguji metode REST API.

Topik

- [Prasyarat](#)
- [Uji metode dengan konsol API Gateway](#)

Prasyarat

- Anda harus menentukan pengaturan untuk metode yang ingin Anda uji. Ikuti petunjuk dalam [Metode untuk REST APIs di API Gateway](#).

Uji metode dengan konsol API Gateway

Important

Metode pengujian dengan konsol API Gateway dapat mengakibatkan perubahan pada sumber daya yang tidak dapat dibatalkan. Menguji metode dengan konsol API Gateway sama dengan memanggil metode di luar konsol API Gateway. Misalnya, jika Anda menggunakan konsol API Gateway untuk memanggil metode yang menghapus sumber daya API, jika pemanggilan metode berhasil, sumber daya API akan dihapus.

Untuk menguji suatu metode

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel Resources, pilih metode yang ingin Anda uji.

4. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

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

Masukkan nilai di salah satu kotak yang ditampilkan (seperti string Kueri, Header, dan badan Permintaan). Konsol menyertakan nilai-nilai ini dalam permintaan metode dalam bentuk aplikasi/json default.

Untuk opsi tambahan yang mungkin perlu Anda tentukan, hubungi pemilik API.

5. Pilih Uji. Informasi berikut akan ditampilkan:

- Permintaan adalah jalur sumber daya yang dipanggil untuk metode.
- Status adalah kode status HTTP respon.
- Latensi (ms) adalah waktu antara penerimaan permintaan dari penelepon dan respons yang dikembalikan.
- Response body adalah badan respon HTTP.
- Response header adalah header respon HTTP.

 Tip

Bergantung pada pemetaan, kode status HTTP, badan respons, dan header respons mungkin berbeda dari yang dikirim dari fungsi Lambda, proxy HTTP, atau proxy layanan. AWS

- Log adalah entri Amazon CloudWatch Log simulasi yang akan ditulis jika metode ini dipanggil di luar konsol API Gateway.

 Note

Meskipun entri CloudWatch Log disimulasikan, hasil pemanggilan metode adalah nyata.

Selain menggunakan konsol API Gateway, Anda dapat menggunakan AWS CLI atau AWS SDK untuk API Gateway untuk menguji pemanggilan metode. Untuk melakukannya dengan menggunakan AWS CLI, lihat [test-invoke-method](#).

Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API

Di bagian ini, kami menguraikan langkah-langkah untuk menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API, dengan menggunakan [Simple Calculator](#) API sebagai contoh. Sebelum melanjutkan, Anda harus menyelesaikan langkah-langkahnya. [Hasilkan SDKs untuk REST APIs di API Gateway](#)

Untuk menginstal dan menggunakan Java SDK yang dihasilkan oleh API Gateway

1. Ekstrak konten file.zip yang dihasilkan API Gateway yang Anda unduh sebelumnya.
2. Unduh dan instal [Apache Maven](#) (harus versi 3.5 atau lebih baru).
3. Unduh dan instal [JDK 8](#).
4. Mengatur variabel JAVA_HOME lingkungan.
5. Buka folder SDK yang tidak di-zip tempat file pom.xml berada. Folder ini secara generated-code default. Jalankan mvn install perintah untuk menginstal file artefak yang dikompilasi ke repositori Maven lokal Anda. Ini membuat target folder yang berisi pustaka SDK yang dikompilasi.
6. Ketik perintah berikut di direktori kosong untuk membuat rintisan proyek klien untuk memanggil API menggunakan pustaka SDK yang diinstal.

```
mvn -B archetype:generate \
    -DarchetypeGroupId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

 Note

Pemisah \ dalam perintah sebelumnya disertakan untuk keterbacaan. Seluruh perintah harus pada satu baris tanpa pemisah.

Perintah ini menciptakan sebuah rintisan aplikasi. Aplikasi rintisan berisi `pom.xml` file dan `src` folder di bawah direktori root proyek (*SimpleCalc-sdkClient* dalam perintah sebelumnya). Awalnya, ada dua file sumber: `src/main/java/{package-path}/App.java` dan `src/test/java/{package-path}/AppTest.java`. Dalam contoh ini, *{package-path}* adalah `examples/aws/apig/simpleCalc/sdk/app`. Jalur paket ini berasal dari `DarchetypeGroupId` nilai. Anda dapat menggunakan `App.java` file sebagai template untuk aplikasi klien Anda, dan Anda dapat menambahkan yang lain di folder yang sama jika diperlukan. Anda dapat menggunakan `AppTest.java` file sebagai templat pengujian unit untuk aplikasi Anda, dan Anda dapat menambahkan file kode pengujian lainnya ke folder pengujian yang sama sesuai kebutuhan.

7. Perbarui dependensi paket dalam `pom.xml` file yang dihasilkan menjadi berikut, ganti name properti `projectGroupId`, dan Anda `artifactId` `version`, jika perlu:

```
<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

Ketika versi artefak dependen yang lebih baru tidak `aws-java-sdk-core` kompatibel dengan versi yang ditentukan di atas (1.11.94), Anda harus memperbarui `<version>` tag ke versi baru.

8. Selanjutnya, kami menunjukkan cara memanggil API menggunakan SDK dengan `memanggilgetABOp(GetABOPRequest req)`, `getApiRoot(GetApiRootRequest req)`, dan `postApiRoot(PostApiRootRequest req)` metode SDK. Metode ini sesuai

dengan GET `/a/b/op`, GET `?a={x}&b={y}&op={operator}`, dan POST / metode, dengan muatan permintaan `{"a": x, "b": y, "op": "operator"}` API, masing-masing.

Perbarui `App.java` file sebagai berikut:

```
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());
    }

```

```
}  
}
```

Dalam contoh sebelumnya, pengaturan konfigurasi yang digunakan untuk membuat instance klien SDK adalah untuk tujuan ilustrasi dan belum tentu direkomendasikan praktik terbaik. Selain itu, menelepon `sdkClient.shutdown()` adalah opsional, terutama jika Anda memerlukan kontrol yang tepat kapan harus membebaskan sumber daya.

Kami telah menunjukkan pola penting untuk memanggil API menggunakan Java SDK. Anda dapat memperluas instruksi untuk memanggil metode API lainnya.

Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API

Di bagian ini, kami akan menguraikan langkah-langkah untuk menggunakan SDK Android yang dihasilkan oleh API Gateway untuk REST API. Sebelum melangkah lebih jauh, Anda harus sudah menyelesaikan langkah-langkahnya. [Hasilkan SDKs untuk REST APIs di API Gateway](#)

Note

SDK yang dihasilkan tidak kompatibel dengan Android 4.4 dan yang lebih lama. Untuk informasi selengkapnya, lihat [the section called “Catatan penting”](#).

Untuk menginstal dan menggunakan Android SDK yang dihasilkan oleh API Gateway

1. Ekstrak konten file.zip yang dihasilkan API Gateway yang Anda unduh sebelumnya.
2. Unduh dan instal [Apache Maven](#) (lebih disukai versi 3.x).
3. Unduh dan instal [JDK 8](#).
4. Mengatur variabel `JAVA_HOME` lingkungan.
5. Jalankan `mvn install` perintah untuk menginstal file artefak yang dikompilasi ke repositori Maven lokal Anda. Ini membuat `target` folder yang berisi pustaka SDK yang dikompilasi.
6. Salin file SDK (nama yang berasal dari Id Artifact dan Versi Artifact yang Anda tentukan saat membuat SDK, misalnya,) dari `target` folder `simple-calcsdk-1.0.0.jar`, bersama dengan semua pustaka lain dari `target/lib` folder, ke folder proyek Anda. `lib`

Jika Anda menggunakan Android Studio, buat `libs` folder di bawah modul aplikasi klien Anda dan salin `file.jar` yang diperlukan ke folder ini. Verifikasi bahwa bagian dependensi dalam file `gradle` modul berisi yang berikut ini.

```
compile fileTree(include: ['*.jar'], dir: 'libs')
compile fileTree(include: ['*.jar'], dir: 'app/libs')
```

Pastikan tidak ada `file.jar` duplikat yang dideklarasikan.

- Gunakan `ApiClientFactory` class untuk menginisialisasi SDK yang dihasilkan API Gateway-generated. Sebagai contoh:

```
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. Untuk menggunakan penyedia kredensial Amazon Cognito untuk mengotorisasi panggilan ke API Anda, gunakan `ApiClientFactory` kelas untuk meneruskan sekumpulan AWS kredensial menggunakan SDK yang dihasilkan oleh API Gateway, seperti yang ditunjukkan pada contoh berikut.

```
// 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. Untuk menyetel kunci API menggunakan SDK yang dihasilkan API Gateway, gunakan kode yang mirip dengan berikut ini.

```
ApiClientFactory factory = new ApiClientFactory()
    .apiKey("YOUR_API_KEY");
```

Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API

Prosedur berikut menunjukkan cara menggunakan JavaScript SDK yang dihasilkan oleh API Gateway.

Note

Instruksi ini menganggap Anda telah menyelesaikan instruksi di [Hasilkan SDKs untuk REST APIs di API Gateway](#).

Important

Jika API Anda hanya memiliki metode APAPUN yang ditentukan, paket SDK yang dihasilkan tidak akan berisi `apigClient.js` file, dan Anda harus menentukan sendiri metode APAPUN.

Untuk menginstal, memulai dan memanggil JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API

1. Ekstrak konten file.zip buatan API Gateway yang Anda unduh sebelumnya.
2. Aktifkan berbagi sumber daya lintas asal (CORS) untuk semua metode yang akan dipanggil oleh SDK yang dihasilkan oleh API Gateway. Untuk petunjuk, lihat [CORS untuk REST APIs di API Gateway](#).
3. Di halaman web Anda, sertakan referensi ke skrip berikut.

```
<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. Dalam kode Anda, inisialisasi SDK yang dihasilkan oleh API Gateway dengan menggunakan kode yang mirip dengan berikut ini.

```
var apigClient = apigClientFactory.newClient();
```

Untuk menginisialisasi SDK yang dihasilkan oleh API Gateway dengan AWS kredensial, gunakan kode yang mirip dengan berikut ini. Jika Anda menggunakan AWS kredensial, semua permintaan ke API akan ditandatangani.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Untuk menggunakan kunci API dengan SDK yang dihasilkan oleh API Gateway, teruskan kunci API sebagai parameter ke Factory objek dengan menggunakan kode yang mirip dengan berikut ini. Jika Anda menggunakan kunci API, kunci tersebut ditentukan sebagai bagian dari `x-api-key` header dan semua permintaan ke API akan ditandatangani. Ini berarti Anda harus mengatur header `CORS Accept` yang sesuai untuk setiap permintaan.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Panggil metode API di API Gateway dengan menggunakan kode yang mirip dengan berikut ini. Setiap panggilan mengembalikan janji dengan callback sukses dan gagal.

```
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.
  });

```

Di sini, *methodName* dibangun dari jalur sumber daya permintaan metode dan kata kerja HTTP. Untuk SimpleCalc API, metode SDK untuk metode API

1. GET `/?a=...&b=...&op=...`
2. POST `/`
3. GET `/a/b/op`

metode SDK yang sesuai adalah sebagai berikut:

1. `rootGet(params);` // where `params={"a": ..., "b": ..., "op": ...}` is resolved to the query parameters
2. `rootPost(null, body);` // where `body={"a": ..., "b": ..., "op": ...}`
3. `aB0pGet(params);` // where `params={"a": ..., "b": ..., "op": ...}` is resolved to the path parameters

Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API

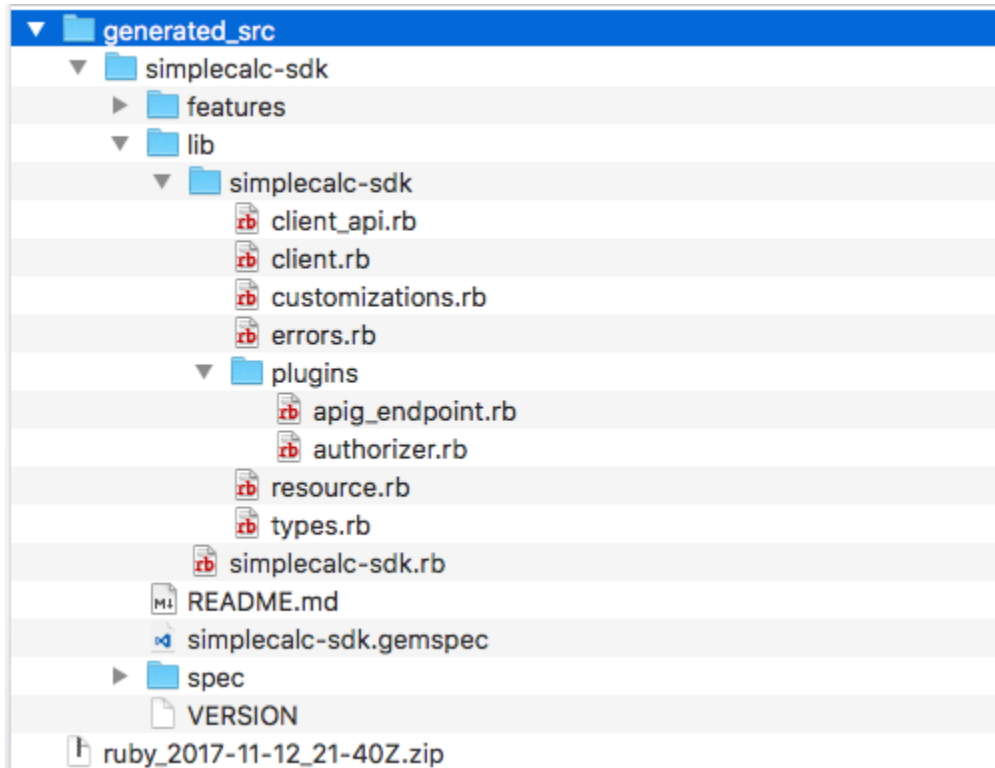
Prosedur berikut menunjukkan cara menggunakan Ruby SDK yang dihasilkan oleh API Gateway.

Note

Instruksi ini mengasumsikan Anda sudah menyelesaikan instruksi di [Hasilkan SDKs untuk REST APIs di API Gateway](#).

Untuk menginstal, membuat instance, dan memanggil Ruby SDK yang dihasilkan oleh API Gateway untuk REST API

1. Buka zip file Ruby SDK yang diunduh. Sumber SDK yang dihasilkan ditampilkan sebagai berikut.



2. Buat Ruby Gem dari sumber SDK yang dihasilkan, menggunakan perintah shell berikut di jendela terminal:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk
```

```
# build the generated gem
gem build simplecalc-sdk.gemspec
```

Setelah ini, simplecalc-sdk-1.0.0.gem menjadi tersedia.

3. Instal pertama:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Buat aplikasi klien. Buat instance dan inisialisasi klien Ruby SDK di aplikasi:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
  http_wire_trace: true,
  retry_limit: 5,
  http_read_timeout: 50
)
```

Jika API memiliki otorisasi AWS_IAM jenis yang dikonfigurasi, Anda dapat menyertakan AWS kredensi pemanggil dengan menyediakan accessKey dan selama inisialisasi: secretKey

```
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. Lakukan panggilan API melalui SDK di aplikasi.

Tip

Jika Anda tidak terbiasa dengan konvensi panggilan metode SDK, Anda dapat meninjau `client.rb` file di folder SDK lib yang dihasilkan. Folder berisi dokumentasi setiap panggilan metode API yang didukung.

Untuk menemukan operasi yang didukung:

```
# to show supported operations:
puts client.operation_names
```

Ini menghasilkan tampilan berikut, sesuai dengan metode API GET /?

a={.}&b={.}&op={.}GET /{a}/{b}/{op}, dan POST /, ditambah muatan {a:"...", b:"...", op:"..."} format, masing-masing:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Untuk menjalankan metode GET /?a=1&b=2&op=+ API, panggil metode Ruby SDK berikut ini:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Untuk menjalankan metode POST / API dengan payload sebesar {a: "1", b: "2", "op": "+"}, panggil metode Ruby SDK berikut:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Untuk menjalankan metode GET /1/2/+ API, panggil metode Ruby SDK berikut:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Pemanggilan metode SDK yang berhasil mengembalikan respons berikut:

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

Menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift

Dalam tutorial ini, kami akan menunjukkan cara menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di aplikasi Objective-C atau Swift untuk memanggil API yang mendasarinya. Kami akan menggunakan [SimpleCalc API](#) sebagai contoh untuk mengilustrasikan topik-topik berikut:

- Cara menginstal komponen AWS Mobile SDK yang diperlukan ke dalam proyek Xcode Anda
- Cara membuat objek klien API sebelum memanggil metode API
- Cara memanggil metode API melalui metode SDK yang sesuai pada objek klien API
- Cara menyiapkan input metode dan mengurai hasilnya menggunakan kelas model SDK yang sesuai

Topik

- [Menggunakan SDK iOS \(Objective-C\) yang dihasilkan untuk memanggil API](#)
- [Menggunakan SDK iOS \(Swift\) yang dihasilkan untuk memanggil API](#)

Menggunakan SDK iOS (Objective-C) yang dihasilkan untuk memanggil API

Sebelum memulai prosedur berikut, Anda harus menyelesaikan langkah-langkah [Hasilkan SDKs untuk REST APIs di API Gateway](#) untuk iOS di Objective-C dan mengunduh file.zip dari SDK yang dihasilkan.

Instal SDK AWS seluler dan SDK iOS yang dihasilkan oleh API Gateway dalam proyek Objective-C

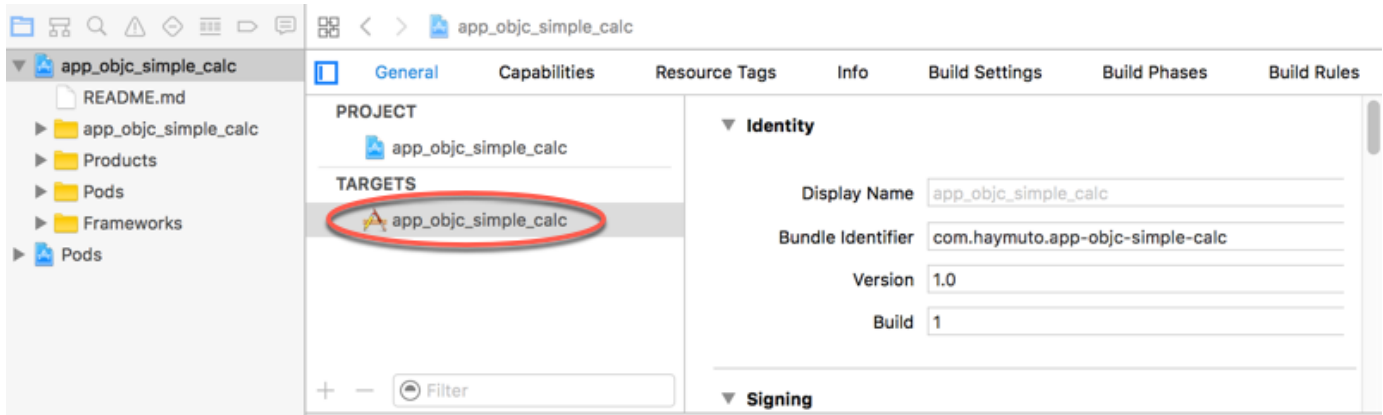
Prosedur berikut menjelaskan cara menginstal SDK.

Untuk menginstal dan menggunakan SDK iOS yang dihasilkan oleh API Gateway di Objective-C

1. Ekstrak konten file.zip buatan API Gateway yang Anda unduh sebelumnya. Dengan menggunakan [SimpleCalc API](#), Anda mungkin ingin mengganti nama folder SDK yang tidak di-zip menjadi sesuatu seperti **sdk_objc_simple_calc**. Di folder SDK ini ada README.md file dan Podfile file. README.md file berisi instruksi untuk menginstal dan menggunakan SDK. Tutorial ini memberikan rincian tentang instruksi ini. Penginstalan memanfaatkan [CocoaPods](#) untuk mengimpor pustaka API Gateway yang diperlukan dan komponen AWS Mobile SDK dependen lainnya. Anda harus memperbarui Podfile untuk mengimpor SDKs ke proyek

Xcode aplikasi Anda. Folder SDK yang tidak diarsipkan juga berisi generated-src folder yang berisi kode sumber SDK yang dihasilkan dari API Anda.

2. Luncurkan Xcode dan buat proyek iOS Objective-C baru. Catat target proyek. Anda harus mengaturnya diPodfile.



3. Untuk mengimpor AWS Mobile SDK for iOS ke dalam proyek Xcode dengan menggunakan CocoaPods, lakukan hal berikut:

- a. Instal CocoaPods dengan menjalankan perintah berikut di jendela terminal:

```
sudo gem install cocoapods
pod setup
```

- b. Salin Podfile file dari folder SDK yang diekstrak ke direktori yang sama yang berisi file proyek Xcode Anda. Ganti blok berikut:

```
target '<YourXcodeTarget>' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

dengan nama target proyek Anda:

```
target 'app_objc_simple_calc' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Jika proyek Xcode Anda sudah berisi file bernamaPodfile, tambahkan baris kode berikut ke dalamnya:

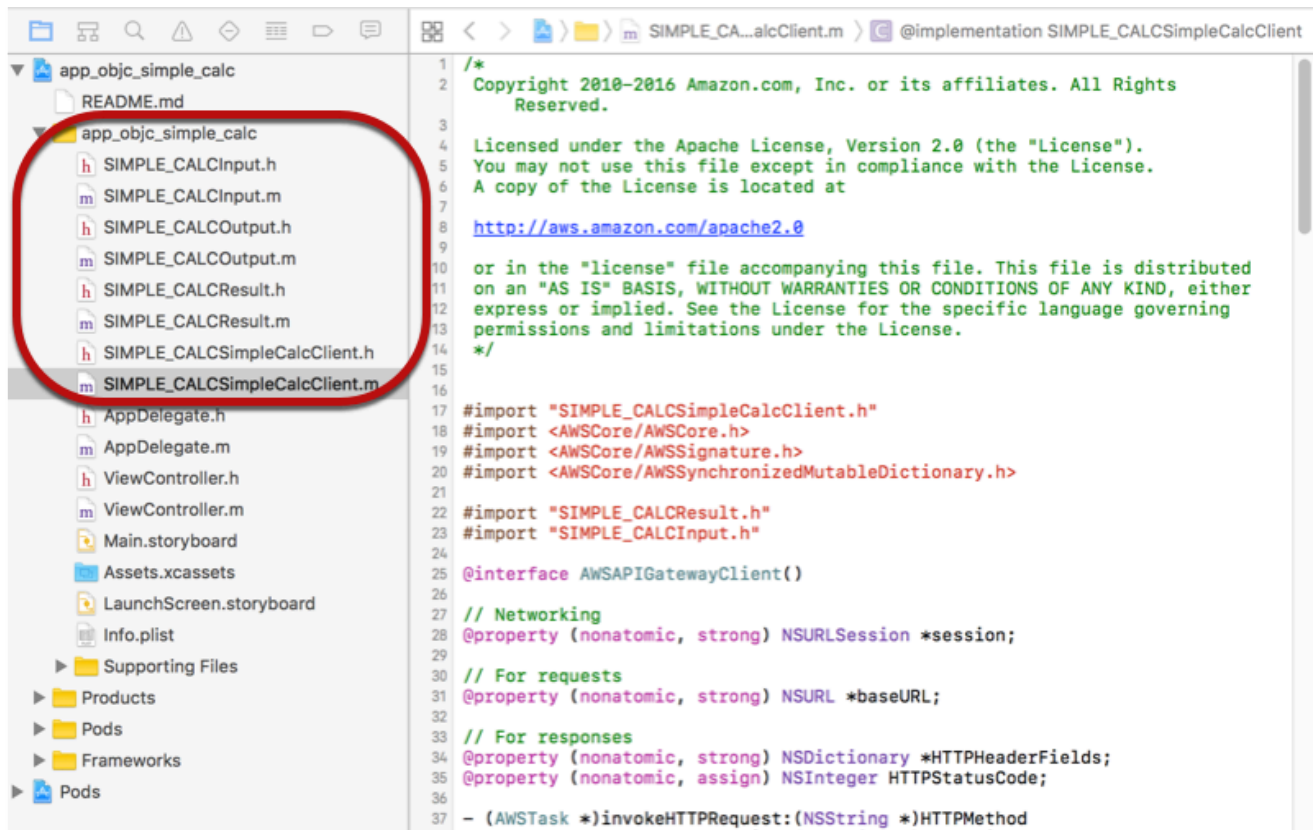
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

- c. Buka jendela terminal dan jalankan perintah berikut:

```
pod install
```

Ini menginstal komponen API Gateway dan komponen AWS Mobile SDK dependen lainnya.

- d. Tutup proyek Xcode dan kemudian buka .xcworkspace file untuk meluncurkan kembali Xcode.
- e. Tambahkan semua .m file .h dan file dari generated-src direktori SDK yang diekstrak ke proyek Xcode Anda.



Untuk mengimpor AWS Mobile SDK for iOS Objective-C ke project Anda dengan mengunduh AWS Mobile SDK secara eksplisit atau menggunakan [Carthage](#), ikuti petunjuk dalam file README.md. Pastikan untuk hanya menggunakan salah satu opsi ini untuk mengimpor SDK AWS Seluler.

Memanggil metode API menggunakan SDK iOS yang dihasilkan oleh API Gateway dalam proyek Objective-C

Saat Anda membuat SDK dengan awalan SIMPLE_CALC untuk [SimpleCalc API](#) ini dengan dua model untuk input (Input) dan output (Result) metode, di SDK, kelas klien API yang dihasilkan menjadi SIMPLE_CALCSimpleCalcClient dan kelas data yang sesuai adalah SIMPLE_CALCInput danSIMPLE_CALCResult, masing-masing. Permintaan dan respons API dipetakan ke metode SDK sebagai berikut:

- Permintaan API dari

```
GET /?a=...&b=...&op=...
```

menjadi metode SDK dari

```
(AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
```

`AWSTask.resultProperty` adalah `SIMPLE_CALCResult` tipe jika `Result` model ditambahkan ke respons metode. Jika tidak, properti adalah `NSDictionary` tipe.

- Permintaan API ini dari

```
POST /  
  
{  
  "a": "Number",  
  "b": "Number",  
  "op": "String"  
}
```

menjadi metode SDK dari

```
(AWSTask *)rootPost:(SIMPLE_CALCInput *)body
```

- Permintaan API dari

```
GET /{a}/{b}/{op}
```

menjadi metode SDK dari


```
(AWSTask *)aB0pGet:(NSString *)a b:(NSString *)b op:(NSString *)op
```

Prosedur berikut menjelaskan cara memanggil metode API dalam kode sumber aplikasi Objective-C; misalnya, sebagai bagian dari viewDidLoad delegasi dalam file. ViewController.m

Untuk memanggil API melalui SDK iOS yang dihasilkan oleh API Gateway

1. Impor file header kelas klien API untuk membuat kelas klien API dapat dipanggil di aplikasi:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

#import Pernyataan itu juga mengimpor SIMPLE_CALCInput.h dan SIMPLE_CALCResult.h untuk dua kelas model.

2. Buat instance class klien API:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
    defaultClient];
```

Untuk menggunakan Amazon Cognito dengan API, setel defaultServiceConfiguration properti pada AWSServiceManager objek default, seperti yang ditunjukkan di bawah ini, sebelum memanggil defaultClient metode untuk membuat objek klien API (ditunjukkan pada contoh sebelumnya):

```
AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
    initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
    configuration;
```

3. Panggil GET /?a=1&b=2&op=+ metode untuk melakukan 1+2:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
    * _Nonnull task) {
    _textField1.text = [self handleApiResponse:task];
    return nil;
}];
```

di mana fungsi pembantu `handleApiResponse:task` memformat hasilnya sebagai string yang akan ditampilkan dalam bidang teks (`_textField1`).

```
- (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;
}
```

Tampilan yang dihasilkan adalah $1 + 2 = 3$.

4. Panggil POST / dengan muatan untuk melakukan $1-2$:

```
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;
}];
```

Tampilan yang dihasilkan adalah $1 - 2 = -1$.

5. Panggil GET /{a}/{b}/{op} untuk melakukan $1/2$:

```
[[apiInstance aB0pGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
_Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

Tampilan yang dihasilkan adalah $1 \text{ div } 2 = 0.5$. Di sini, `div` digunakan sebagai pengganti / karena [fungsi Lambda sederhana](#) di backend tidak menangani variabel jalur yang dikodekan URL.

Menggunakan SDK iOS (Swift) yang dihasilkan untuk memanggil API

Sebelum memulai prosedur berikut, Anda harus menyelesaikan langkah-langkah [Hasilkan SDKs untuk REST APIs di API Gateway](#) untuk iOS di Swift dan mengunduh file.zip dari SDK yang dihasilkan.

Topik

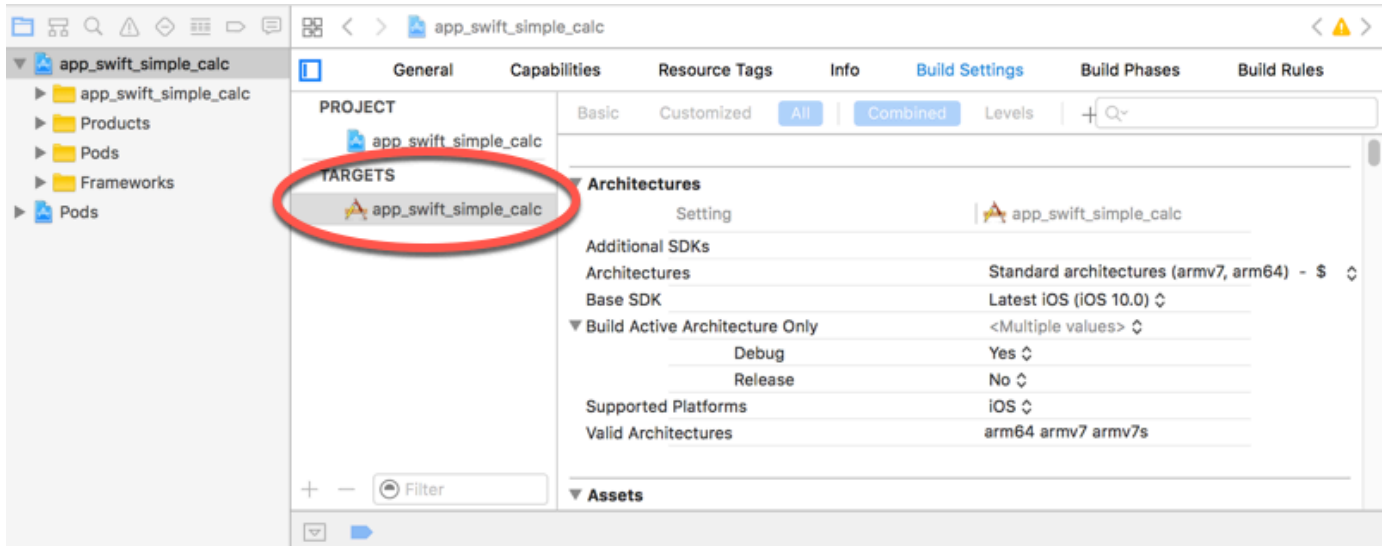
- [Instal SDK AWS seluler dan SDK yang dihasilkan API Gateway dalam proyek Swift](#)
- [Memanggil metode API melalui SDK iOS yang dihasilkan oleh API Gateway dalam proyek Swift](#)

Instal SDK AWS seluler dan SDK yang dihasilkan API Gateway dalam proyek Swift

Prosedur berikut menjelaskan cara menginstal SDK.

Untuk menginstal dan menggunakan SDK iOS yang dihasilkan oleh API Gateway di Swift

1. Ekstrak konten file.zip buatan API Gateway yang Anda unduh sebelumnya. Dengan menggunakan [SimpleCalc API](#), Anda mungkin ingin mengganti nama folder SDK yang tidak di-zip menjadi sesuatu seperti. **sdk_swift_simple_calc** Di folder SDK ini ada README.md file dan Podfile file. README.md File berisi instruksi untuk menginstal dan menggunakan SDK. Tutorial ini memberikan rincian tentang instruksi ini. Penginstalan memanfaatkan [CocoaPods](#) untuk mengimpor komponen AWS Mobile SDK yang diperlukan. Anda harus memperbarui Podfile untuk mengimpor SDKs ke proyek Xcode aplikasi Swift Anda. Folder SDK yang tidak diarsipkan juga berisi generated-src folder yang berisi kode sumber SDK yang dihasilkan dari API Anda.
2. Luncurkan Xcode dan buat proyek iOS Swift baru. Catat target proyek. Anda harus mengaturnya diPodfile.



3. Untuk mengimpor komponen AWS Mobile SDK yang diperlukan ke dalam proyek Xcode dengan menggunakan CocoaPods, lakukan hal berikut:

- a. Jika tidak diinstal, instal CocoaPods dengan menjalankan perintah berikut di jendela terminal:

```
sudo gem install cocoapods
pod setup
```

- b. Salin Podfile file dari folder SDK yang diekstrak ke direktori yang sama yang berisi file proyek Xcode Anda. Ganti blok berikut:

```
target '<YourXcodeTarget>' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

dengan nama target proyek Anda seperti yang ditunjukkan:

```
target 'app_swift_simple_calc' do
  pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Jika proyek Xcode Anda sudah berisi a Podfile dengan target yang benar, Anda cukup menambahkan baris kode berikut ke `do ... end` loop:

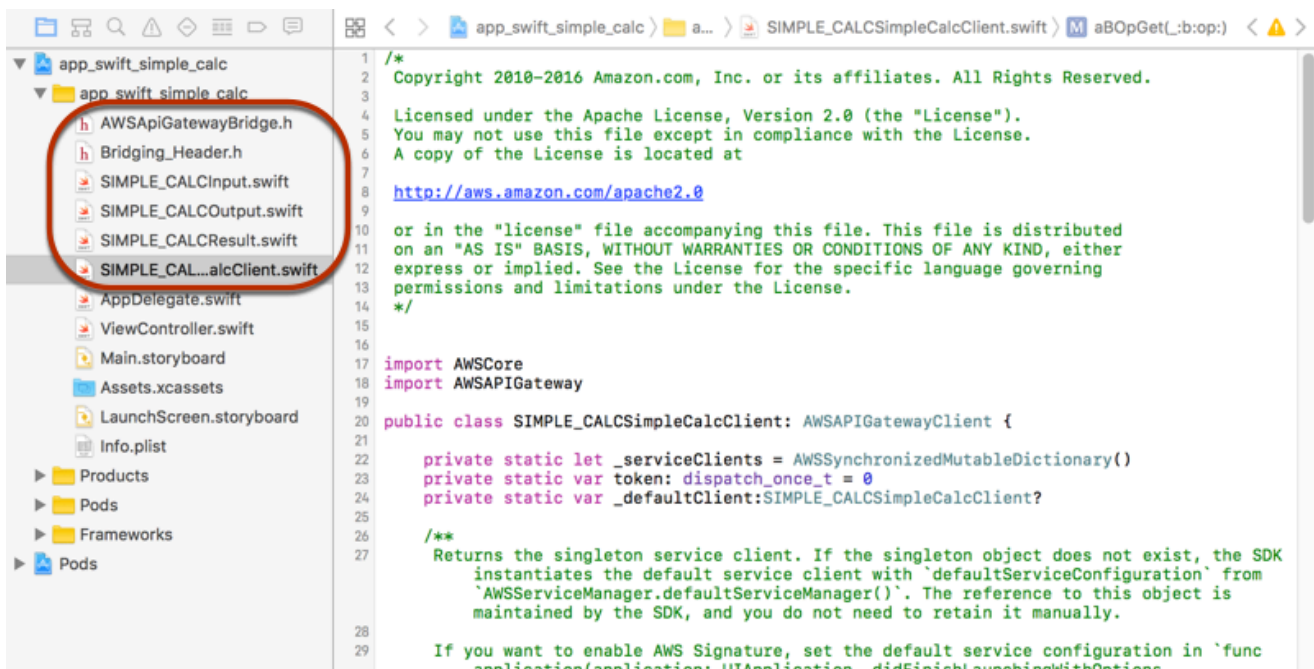
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

- c. Buka jendela terminal dan jalankan perintah berikut di direktori aplikasi:

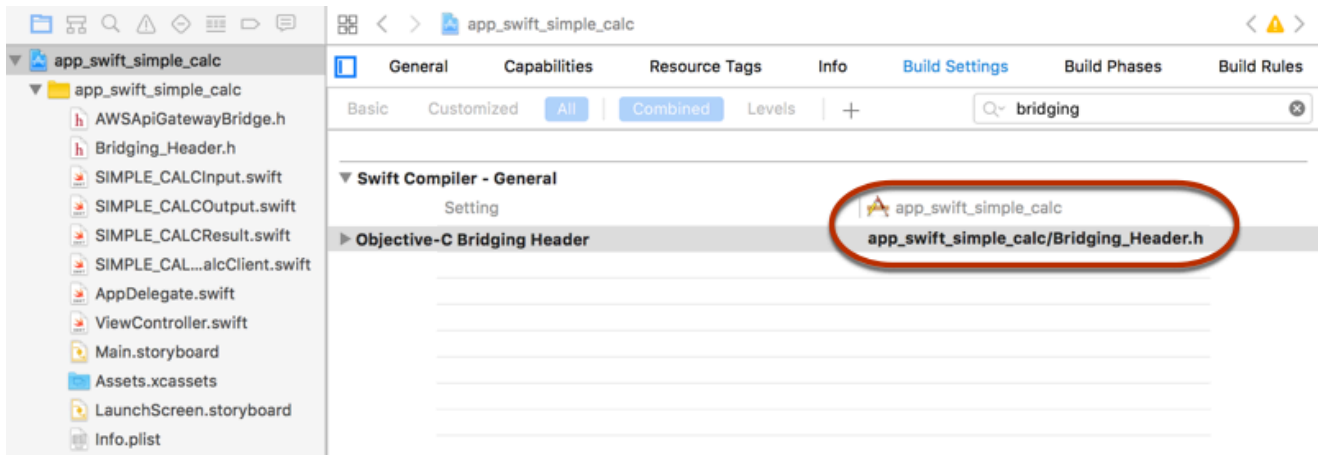
```
pod install
```

Ini menginstal komponen API Gateway dan komponen AWS Mobile SDK dependen apa pun ke dalam proyek aplikasi.

- d. Tutup proyek Xcode dan kemudian buka *.xcworkspace file untuk meluncurkan kembali Xcode.
- e. Tambahkan semua file header SDK (.h) dan file kode sumber Swift (.swift) dari generated-src direktori yang diekstrak ke proyek Xcode Anda.



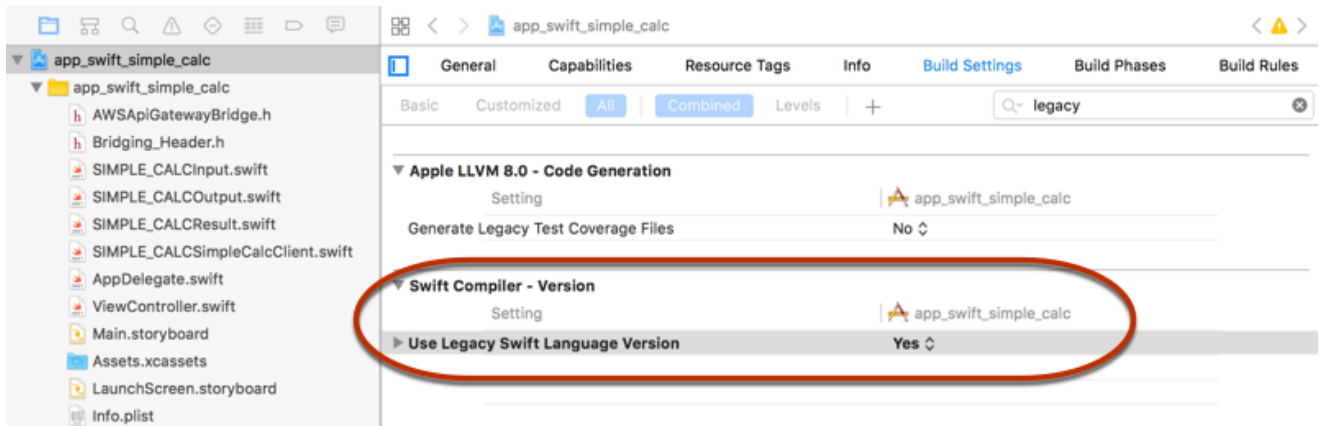
- f. Untuk mengaktifkan pemanggilan pustaka Objective-C dari AWS Mobile SDK dari proyek kode Swift Anda, setel path **Bridging_Header.h** file pada properti Objective-C Bridging Header di bawah Swift Compiler - Pengaturan umum konfigurasi proyek Xcode Anda:



Tip

Anda dapat mengetik **bridging** di kotak pencarian Xcode untuk menemukan properti Objective-C Bridging Header.

- g. Bangun proyek Xcode untuk memverifikasi bahwa itu dikonfigurasi dengan benar sebelum melanjutkan lebih jauh. Jika Xcode Anda menggunakan versi Swift yang lebih baru daripada yang didukung untuk AWS Mobile SDK, Anda akan mendapatkan kesalahan kompil Swift. Dalam hal ini, setel properti Use Legacy Swift Language Version ke Ya di bawah pengaturan Swift Compiler - Version:



Untuk mengimpor SDK AWS Seluler untuk iOS di Swift ke proyek Anda dengan mengunduh AWS SDK Seluler secara eksplisit atau [menggunakan](#) Carthage, ikuti petunjuk dalam README.md file yang disertakan dengan paket SDK. Pastikan untuk hanya menggunakan salah satu opsi ini untuk mengimpor SDK AWS Seluler.

Memanggil metode API melalui SDK iOS yang dihasilkan oleh API Gateway dalam proyek Swift

Saat Anda membuat SDK dengan awalan untuk [SimpleCalc API](#) ini dengan dua model `SIMPLE_CALC` untuk mendeskripsikan input (Input) dan output (Result) permintaan dan respons API, di SDK, kelas klien API yang dihasilkan menjadi `SIMPLE_CALCSimpleCalcClient` dan kelas data yang sesuai adalah `SIMPLE_CALCInput` dan `SIMPLE_CALCResult`, masing-masing. Permintaan dan respons API dipetakan ke metode SDK sebagai berikut:

- Permintaan API dari

```
GET /?a=...&b=...&op=...
```

menjadi metode SDK dari

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

`AWSTask.resultProperiti` adalah `SIMPLE_CALCResult` tipe jika `Result` model ditambahkan ke respons metode. Kalau tidak, itu adalah `NSDictionary` tipe.

- Permintaan API ini dari

```
POST /  
  
{  
  "a": "Number",  
  "b": "Number",  
  "op": "String"  
}
```

menjadi metode SDK dari

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- Permintaan API dari

```
GET /{a}/{b}/{op}
```

menjadi metode SDK dari

```
public func aB0pGet(a: String, b: String, op: String) -> AWSTask
```

Prosedur berikut menjelaskan cara memanggil metode API dalam kode sumber aplikasi Swift; misalnya, sebagai bagian dari `viewDidLoad()` delegasi dalam file `ViewController.m`

Untuk memanggil API melalui SDK iOS yang dihasilkan oleh API Gateway

1. Buat instance class klien API:

```
let client = SIMPLE_CALCSimpleCalcClient.default()
```

Untuk menggunakan Amazon Cognito dengan API, tetapkan konfigurasi AWS layanan default (ditampilkan berikut) sebelum mendapatkan default metode (ditampilkan sebelumnya):

```
let credentialsProvider =
    AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
        "my_pool_id")
let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
    credentialsProvider: credentialsProvider)
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
    configuration
```

2. Panggil GET `/?a=1&b=2&op=+` metode untuk melakukan `1+2`:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
    in
        self.showResult(task)
        return nil
}
```

di mana fungsi pembantu `self.showResult(task)` mencetak hasil atau kesalahan ke konsol; misalnya:

```
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)")
    }
}
}

```

Di aplikasi produksi, Anda dapat menampilkan hasil atau kesalahan di bidang teks. Tampilan yang dihasilkan adalah $1 + 2 = 3$.

3. Panggil POST / dengan muatan untuk melakukan $1-2$:

```

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
}

```

Tampilan yang dihasilkan adalah $1 - 2 = -1$.

4. Panggil GET /{a}/{b}/{op} untuk melakukan $1/2$:

```

client.aB0pGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
AnyObject? in
    self.showResult(task)
    return nil
}

```

Tampilan yang dihasilkan adalah $1 \text{ div } 2 = 0.5$. Di sini, `div` digunakan sebagai pengganti / karena [fungsi Lambda sederhana](#) di backend tidak menangani variabel jalur yang dikodekan URL.

Kembangkan REST API menggunakan OpenAPI di API Gateway

Anda dapat menggunakan API Gateway untuk mengimpor REST API dari file definisi eksternal ke API Gateway. Saat ini, API Gateway mendukung file definisi [OpenAPIv2.0](#) dan [OpenAPIv3.0](#),

dengan pengecualian yang tercantum dalam file. [Catatan penting Amazon API Gateway untuk REST APIs](#) Anda dapat memperbarui API dengan menimpa dengan definisi baru, atau Anda dapat menggabungkan definisi dengan API yang ada. Anda menentukan opsi dengan menggunakan parameter mode kueri di URL permintaan.

Untuk tutorial tentang penggunaan fitur Impor API dari konsol API Gateway, lihat [Tutorial: Buat REST API dengan mengimpor contoh](#).

Topik

- [Impor API yang dioptimalkan tepi ke API Gateway](#)
- [Impor API Regional ke API Gateway](#)
- [Impor file OpenAPI untuk memperbarui definisi API yang ada](#)
- [Mengatur properti BasePath OpenAPI](#)
- [AWS variabel untuk impor OpenAPI](#)
- [Kesalahan dan peringatan dari mengimpor API Anda ke API Gateway](#)
- [Ekspor REST API dari API Gateway](#)

Impor API yang dioptimalkan tepi ke API Gateway

Anda dapat mengimpor file definisi OpenAPI API untuk membuat API baru yang dioptimalkan tepi dengan menentukan tipe EDGE titik akhir sebagai input tambahan, selain file OpenAPI, ke operasi impor. Anda dapat melakukannya menggunakan konsol API Gateway AWS CLI, atau AWS SDK.

Untuk tutorial tentang penggunaan fitur Impor API dari konsol API Gateway, lihat [Tutorial: Buat REST API dengan mengimpor contoh](#).

Topik

- [Impor API yang dioptimalkan tepi menggunakan konsol API Gateway](#)
- [Impor API yang dioptimalkan tepi menggunakan AWS CLI](#)

Impor API yang dioptimalkan tepi menggunakan konsol API Gateway

Untuk mengimpor API yang dioptimalkan tepi menggunakan konsol API Gateway, lakukan hal berikut:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Pilih Buat API.
3. Di bawah REST API, pilih Impor.
4. Salin definisi OpenAPI API dan tempelkan ke editor kode, atau pilih Pilih file untuk memuat file OpenAPI dari drive lokal.
5. Untuk jenis titik akhir API, pilih Optimalisasi tepi.
6. Pilih Buat API untuk mulai mengimpor definisi OpenAPI.

Impor API yang dioptimalkan tepi menggunakan AWS CLI

[import-rest-api](#) Perintah berikut mengimpor API dari file definisi OpenAPI untuk membuat API baru yang dioptimalkan tepi:

```
aws apigateway import-rest-api \  
  --fail-on-warnings \  
  --body 'file://path/to/API_OpenAPI_template.json'
```

atau dengan spesifikasi eksplisit dari parameter string `endpointConfigurationTypes` kueri untuk `EDGE`:

```
aws apigateway import-rest-api \  
  --parameters endpointConfigurationTypes=EDGE \  
  --fail-on-warnings \  
  --body 'file://path/to/API_OpenAPI_template.json'
```

Impor API Regional ke API Gateway

Saat mengimpor API, Anda dapat memilih konfigurasi titik akhir regional untuk API. Anda dapat menggunakan konsol API Gateway, the AWS CLI, atau AWS SDK.

Saat Anda mengekspor API, konfigurasi titik akhir API tidak disertakan dalam definisi API yang diekspor.

Untuk tutorial tentang penggunaan fitur Impor API dari konsol API Gateway, lihat [Tutorial: Buat REST API dengan mengimpor contoh](#).

Topik

- [Mengimpor API regional menggunakan konsol API Gateway](#)

- [Impor API regional menggunakan AWS CLI](#)

Mengimpor API regional menggunakan konsol API Gateway

Untuk mengimpor API titik akhir regional menggunakan konsol API Gateway, lakukan hal berikut:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API.
3. Di bawah REST API, pilih Impor.
4. Salin definisi OpenAPI API dan tempelkan ke editor kode, atau pilih Pilih file untuk memuat file OpenAPI dari drive lokal.
5. Untuk jenis titik akhir API, pilih Regional.
6. Pilih Buat API untuk mulai mengimpor definisi OpenAPI.

Impor API regional menggunakan AWS CLI

[import-rest-api](#) Perintah berikut mengimpor file definisi OpenAPI dan menetapkan tipe endpoint ke Regional:

```
aws apigateway import-rest-api \  
  --parameters endpointConfigurationTypes=REGIONAL \  
  --fail-on-warnings \  
  --body 'file://path/to/API_OpenAPI_template.json'
```

Impor file OpenAPI untuk memperbarui definisi API yang ada

Anda dapat mengimpor definisi API hanya untuk memperbarui API yang ada, tanpa mengubah konfigurasi titik akhir, serta variabel tahapan dan tahap, atau referensi ke kunci API.

Operasi impor-untuk-pembaruan dapat terjadi dalam dua mode: menggabungkan atau menimpa.

Ketika API (A) digabungkan ke lain (B), API yang dihasilkan mempertahankan definisi keduanya A dan B jika kedua API tidak memiliki definisi yang bertentangan. Ketika konflik muncul, definisi metode dari API penggabungan (A) akan mengesampingkan definisi metode yang sesuai dari API gabungan (). B Misalnya, misalkan B telah menyatakan metode berikut untuk kembali 200 dan 206 tanggapan:

```
GET /a
```

```
POST /a
```

dan A mendeklarasikan metode berikut untuk kembali 200 dan 400 tanggapan:

```
GET /a
```

Saat A digabungkanB, API yang dihasilkan menghasilkan metode berikut:

```
GET /a
```

yang kembali 200 dan 400 tanggapan, dan

```
POST /a
```

yang kembali 200 dan 206 tanggapan.

Menggabungkan API berguna ketika Anda telah menguraikan definisi API eksternal Anda menjadi beberapa bagian yang lebih kecil dan hanya ingin menerapkan perubahan dari salah satu bagian tersebut pada satu waktu. Misalnya, ini mungkin terjadi jika beberapa tim bertanggung jawab atas bagian API yang berbeda dan memiliki perubahan yang tersedia dengan tarif yang berbeda. Dalam mode ini, item dari API yang ada yang tidak ditentukan secara khusus dalam definisi yang diimpor dibiarkan saja.

Saat API (A) menimpa API (B) lain, API yang dihasilkan mengambil definisi dari API penimpaan (A). Menimpa API berguna ketika definisi API eksternal berisi definisi lengkap API. Dalam mode ini, item dari API yang ada yang tidak ditentukan secara khusus dalam definisi yang diimpor akan dihapus.

Important

Saat Anda menggunakan `mode=overwrite`, seluruh definisi API diganti. Konfigurasi API apa pun yang tidak secara eksplisit disertakan dalam badan impor disetel ulang ke nilai defaultnya. Ini termasuk pengaturan seperti kebijakan keamanan, konfigurasi otorisasi, dan properti lainnya API-level .

Untuk mempertahankan konfigurasi yang ada, gunakan `mode=merge`, yang mempertahankan konfigurasi yang ada yang tidak ditentukan dalam badan impor. Jika Anda menggunakan `mode=overwrite`, pastikan bahwa definisi OpenAPI Anda mencakup semua konfigurasi API yang diinginkan, seperti ekstensi API Gateway.

Untuk menggabungkan API, kirimkan PUT permintaan ke `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. Nilai parameter `restapi_id` jalur menentukan API tempat definisi API yang disediakan akan digabungkan.

Cuplikan kode berikut menunjukkan contoh PUT permintaan untuk menggabungkan definisi OpenAPI API di JSON, sebagai payload, dengan API tertentu yang sudah ada di API Gateway.

```
PUT /restapis/<restapi_id>?mode=merge
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...
```

[An OpenAPI API definition in JSON](#)

Operasi pembaruan penggabungan mengambil dua definisi API lengkap dan menggabungkannya bersama-sama. Untuk perubahan kecil dan bertahap, Anda dapat menggunakan operasi [pembaruan sumber daya](#).

Untuk menimpa API, kirimkan PUT permintaan ke `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. Parameter `restapi_id` path menentukan API yang akan ditimpa dengan definisi API yang disediakan.

Cuplikan kode berikut menunjukkan contoh permintaan penyimpanan dengan muatan definisi: 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](#)

Ketika parameter mode kueri tidak ditentukan, penggabungan diasumsikan.

 Note

PUT Operasi itu idempoten, tetapi tidak atom. Itu berarti jika kesalahan sistem terjadi sebagian melalui pemrosesan, API dapat berakhir dalam keadaan buruk. Namun, mengulangi operasi berhasil menempatkan API ke status akhir yang sama seolah-olah operasi pertama telah berhasil.

Mengatur properti BasePath OpenAPI

Di [OpenAPI 2.0](#), Anda dapat menggunakan basePath properti untuk menyediakan satu atau beberapa bagian jalur yang mendahului setiap jalur yang ditentukan dalam properti. paths Karena API Gateway memiliki beberapa cara untuk mengekspresikan jalur sumber daya, fitur Import API menyediakan opsi berikut untuk menafsirkan basePath properti selama import: ignore, prepend, dan split.

Di [OpenAPI 3.0](#), basePath bukan lagi properti tingkat atas. Sebagai gantinya, API Gateway menggunakan [variabel server](#) sebagai konvensi. Fitur Import API menyediakan opsi yang sama untuk menafsirkan jalur dasar selama impor. Jalur dasar diidentifikasi sebagai berikut:

- Jika API tidak berisi basePath variabel apa pun, fitur Import API akan memeriksa `server.url` string untuk melihat apakah itu berisi jalur di luar `"/`. Jika ya, jalur itu digunakan sebagai jalur dasar.
- Jika API hanya berisi satu basePath variabel, fitur Import API menggunakannya sebagai jalur dasar, meskipun tidak direferensikan `server.url` dalam.
- Jika API berisi beberapa basePath variabel, fitur Import API hanya menggunakan yang pertama sebagai jalur dasar.

Abaikan

Jika file OpenAPI memiliki basePath nilai `/a/b/c` dan paths properti berisi `/e` dan `/f`, berikut POST atau PUT permintaan:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

menghasilkan sumber daya berikut di API:

- /
- /e
- /f

Efeknya adalah memperlakukan basePath seolah-olah tidak ada, dan semua sumber daya API yang dideklarasikan disajikan relatif terhadap host. Ini dapat digunakan, misalnya, ketika Anda memiliki nama domain khusus dengan pemetaan API yang tidak menyertakan Jalur Dasar dan nilai Tahap yang mengacu pada tahap produksi Anda.

 Note

API Gateway secara otomatis membuat sumber daya root untuk Anda, meskipun tidak dinyatakan secara eksplisit dalam file definisi Anda.

Ketika tidak ditentukan, basePath mengambil secara ignore default.

Prepend

Jika OpenAPI file memiliki basePath nilai /a/b/c dan paths properti berisi /e dan /f, berikut POST atau PUT permintaan:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

menghasilkan sumber daya berikut di API:

- /
- /a
- /a/b
- /a/b/c
- /a/b/c/e

- /a/b/c/f

Efeknya adalah memperlakukan basePath sebagai menentukan sumber daya tambahan (tanpa metode) dan menambahkannya ke kumpulan sumber daya yang dideklarasikan. Ini dapat digunakan, misalnya, ketika tim yang berbeda bertanggung jawab atas bagian API yang berbeda dan basePath dapat mereferensikan lokasi jalur untuk setiap bagian API tim.

 Note

API Gateway secara otomatis membuat sumber daya perantara untuk Anda, meskipun sumber daya tersebut tidak dideklarasikan secara eksplisit dalam definisi Anda.

Split

Jika OpenAPI file memiliki basePath nilai /a/b/c dan paths properti berisi /e dan/f, berikut POST atau PUT permintaan:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

menghasilkan sumber daya berikut di API:

- /
- /b
- /b/c
- /b/c/e
- /b/c/f

Efeknya adalah memperlakukan bagian jalur paling atas, /a, sebagai awal dari setiap jalur sumber daya, dan untuk membuat sumber daya tambahan (tanpa metode) dalam API itu sendiri. Ini dapat, misalnya, digunakan ketika a merupakan nama panggung yang ingin Anda ekspos sebagai bagian dari API Anda.

AWS variabel untuk impor OpenAPI

Anda dapat menggunakan AWS variabel berikut dalam definisi OpenAPI. API Gateway menyelesaikan variabel saat API diimpor. Untuk menentukan variabel, gunakan `${variable-name}`. Tabel berikut menjelaskan AWS variabel yang tersedia.

Nama variabel	Deskripsi	
AWS::AccountId	ID AWS akun yang mengimpor API. Misalnya, 123456789012.	
AWS::Partition	AWS Partisi di mana API diimpor. Untuk AWS Wilayah standar, partisi adalah <code>aws</code> .	
AWS::Region	AWS Wilayah di mana API diimpor. Misalnya, <code>us-east-2</code> .	

AWS variabel contoh

Contoh berikut menggunakan AWS variabel untuk menentukan AWS Lambda fungsi untuk integrasi.

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

```

Kesalahan dan peringatan dari mengimpor API Anda ke API Gateway

Saat Anda mengimpor file definisi eksternal ke API Gateway, API Gateway mungkin menghasilkan peringatan dan kesalahan. Bagian berikut membahas kesalahan dan peringatan yang mungkin terjadi selama impor.

Kesalahan selama impor

Selama impor, kesalahan dapat dihasilkan untuk masalah utama seperti dokumen yang tidak validOpenAPI. Kesalahan dikembalikan sebagai pengecualian (misalnya, `BadRequestException`)

dalam respons yang tidak berhasil. Ketika terjadi kesalahan, definisi API baru dibuang dan tidak ada perubahan yang dilakukan pada API yang ada.

Peringatan selama impor

Selama impor, peringatan dapat dibuat untuk masalah kecil seperti referensi model yang hilang. Jika peringatan terjadi, operasi akan dilanjutkan jika ekspresi `failonwarnings=false` kueri ditambahkan ke URL permintaan. Jika tidak, pembaruan akan diputar kembali. Secara default, `failonwarnings` diatur ke `false`. Dalam kasus seperti itu, peringatan dikembalikan sebagai bidang di [RestApi](#) sumber daya yang dihasilkan. Jika tidak, peringatan dikembalikan sebagai pesan dalam pengecualian.

Ekspor REST API dari API Gateway

Setelah Anda membuat dan mengonfigurasi REST API di API Gateway, menggunakan konsol API Gateway atau sebaliknya, Anda dapat mengekspornya ke file OpenAPI menggunakan API Gateway Export API, yang merupakan bagian dari Amazon API Gateway Control Service. Untuk menggunakan API Gateway Export API, Anda harus menandatangani permintaan API Anda. Untuk informasi selengkapnya tentang permintaan [penandatanganan](#), lihat [Menandatangani permintaan AWS API](#) di Panduan Pengguna IAM. Anda memiliki opsi untuk menyertakan ekstensi integrasi API Gateway, serta ekstensi [Postman](#), dalam file definisi OpenAPI yang diekspor.

Note

Saat mengekspor API menggunakan AWS CLI, pastikan untuk menyertakan parameter ekstensi seperti yang ditunjukkan pada contoh berikut, untuk memastikan bahwa `x-amazon-apigateway-request-validator` ekstensi disertakan:

```
aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
```

Anda tidak dapat mengekspor API jika payloadnya bukan dari `application/json` jenisnya. Jika Anda mencoba, Anda akan mendapatkan respons kesalahan yang menyatakan bahwa model tubuh JSON tidak ditemukan.

Permintaan untuk mengekspor REST API

Dengan Export API, Anda mengekspor REST API yang ada dengan mengirimkan permintaan GET, menetapkan to-be-exported API sebagai bagian dari jalur URL. URL permintaan adalah dari format berikut:

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
```

Anda dapat menambahkan string extensions kueri untuk menentukan apakah akan menyertakan ekstensi API Gateway (dengan integration nilai) atau ekstensi Postman (dengan postman nilai).

Selain itu, Anda dapat mengatur Accept header ke application/json atau application/yaml untuk menerima output definisi API masing-masing dalam format JSON atau YAMAL.

Untuk informasi selengkapnya tentang mengirimkan permintaan GET menggunakan API Gateway Export API, lihat. [GetExport](#)

Note

Jika Anda mendefinisikan model di API Anda, model tersebut harus untuk jenis konten “application/json” agar API Gateway dapat mengekspor model. Jika tidak, API Gateway melempar pengecualian dengan pesan kesalahan “Hanya menemukan model tubuh non-JSON untuk...”.

Model harus berisi properti atau didefinisikan sebagai JSONSchema tipe tertentu.

Unduh definisi REST API OpenAPI dalam JSON

Untuk mengekspor dan mengunduh REST API dalam definisi OpenAPI dalam format JSON:

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
```

Di sini, **<region>** bisa jadi, misalnya, `us-east-1`. Untuk semua wilayah di mana API Gateway tersedia, lihat [Wilayah dan Titik Akhir](#).

Unduh definisi REST API OpenAPI dalam YAMAL

Untuk mengeksport dan mengunduh REST API dalam definisi OpenAPI dalam format YAMAL:

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
```

Unduh definisi OpenAPI REST API dengan ekstensi Postman di JSON

Untuk mengeksport dan mengunduh REST API dalam definisi OpenAPI dengan format Postman JSON:

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
```

Unduh definisi OpenAPI REST API dengan integrasi API Gateway di YAMAL

Untuk mengeksport dan mengunduh REST API dalam definisi OpenAPI dengan integrasi API Gateway dalam format YAMAL:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

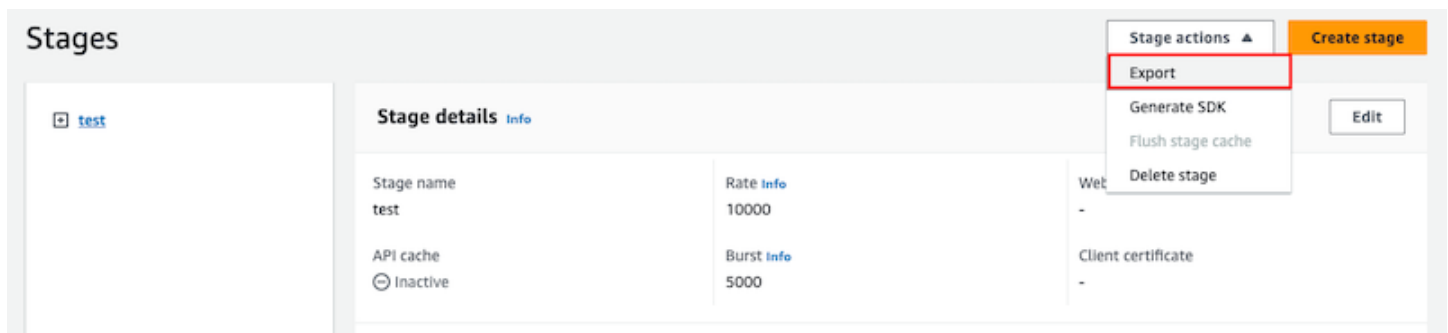
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?  
extensions=integrations  
Host: apigateway.<region>.amazonaws.com  
Accept: application/yaml
```

Ekspor REST API menggunakan konsol API Gateway

Setelah [menerapkan REST API ke tahap](#), Anda dapat melanjutkan untuk mengekspor API di tahap ke file OpenAPI menggunakan konsol API Gateway.

Di panel Tahapan di konsol API Gateway, pilih Tindakan tahap, Ekspor.



Tentukan jenis spesifikasi API, Format, dan Ekstensi untuk mengunduh definisi OpenAPI API Anda.

Publikasikan REST APIs agar pelanggan dapat memanggil

Cukup membuat dan mengembangkan API Gateway API tidak secara otomatis membuatnya dapat dipanggil oleh pengguna Anda. Untuk membuatnya dapat dipanggil, Anda harus menerapkan API Anda ke sebuah panggung. Selain itu, Anda mungkin ingin menyesuaikan URL yang akan digunakan pengguna Anda untuk mengakses API Anda. Anda dapat memberikan domain yang konsisten dengan merek Anda atau lebih mudah diingat daripada URL default untuk API Anda.

Di bagian ini, Anda dapat mempelajari cara menerapkan API dan menyesuaikan URL yang Anda berikan kepada pengguna untuk mengaksesnya.

 Note

Untuk meningkatkan keamanan API Gateway Anda APIs, `execute-api`. `{region}.amazonaws.com` domain terdaftar di [Daftar Akhiran Publik \(PSL\)](#). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway APIs Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman [Set-Cookie](#) di Jaringan Pengembang Mozilla.

Topik

- [Menerapkan REST APIs di API Gateway](#)
- [Nama domain khusus untuk REST publik APIs di API Gateway](#)

Menerapkan REST APIs di API Gateway

Setelah membuat API, Anda harus menerapkannya agar dapat dipanggil oleh pengguna Anda.

Untuk menerapkan API, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung. Stage adalah referensi logis ke status siklus hidup API Anda (misalnya,, dev, prodbeta,v2). Tahapan API diidentifikasi oleh ID API dan nama panggung. Mereka disertakan dalam URL yang Anda gunakan untuk menjalankan API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

 Important

Setiap kali Anda memperbarui API, Anda harus menerapkan ulang API ke tahap yang ada atau ke tahap baru. Memperbarui API mencakup memodifikasi rute, metode, integrasi, otorisasi, kebijakan sumber daya, dan hal lain selain pengaturan tahap.

Saat API Anda berkembang, Anda dapat terus menerapkannya ke tahapan yang berbeda sebagai versi API yang berbeda. Anda juga dapat menerapkan pembaruan API Anda sebagai penerapan [rilis canary](#). Hal ini memungkinkan klien API Anda untuk mengakses, pada tahap yang sama, versi produksi melalui rilis produksi, dan versi terbaru melalui rilis canary.

Untuk memanggil API yang diterapkan, klien mengirimkan permintaan terhadap URL API. URL ditentukan oleh protokol API (HTTP (S) atau (WSS)), nama host, nama panggung, dan jalur sumber daya (untuk REST APIs). Nama host dan nama panggung menentukan URL dasar API.

Menggunakan nama domain default API, URL dasar REST API (misalnya) dalam tahap tertentu (*{stageName}*) adalah dalam format berikut:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Untuk membuat URL dasar default API lebih ramah pengguna, Anda dapat membuat nama domain khusus (misalnya, *api.example.com*) untuk mengganti nama host default API. Untuk mendukung beberapa APIs di bawah nama domain kustom, Anda harus memetakan tahap API ke jalur dasar.

Dengan nama domain kustom *{api.example.com}* dan tahap API yang dipetakan ke jalur dasar (*{basePath}*) di bawah nama domain kustom, URL dasar REST API menjadi sebagai berikut:

```
https://{api.example.com}/{basePath}
```

Untuk setiap tahap, Anda dapat mengoptimalkan performa API dengan menyesuaikan batas pembatasan permintaan tingkat akun default dan mengaktifkan caching API. Anda juga dapat mengaktifkan logging untuk panggilan API ke CloudTrail atau CloudWatch, dan dapat memilih sertifikat klien untuk backend untuk mengautentikasi permintaan API. Selain itu, Anda dapat mengganti setelan tingkat tahap untuk metode individual dan menentukan variabel tahap untuk meneruskan konteks lingkungan khusus tahap ke integrasi API saat runtime.

Tahapan memungkinkan kontrol versi yang kuat dari API Anda. Misalnya, Anda dapat menerapkan API ke test stage dan prod stage, dan menggunakan test stage sebagai test build dan menggunakan prod stage sebagai build stabil. Setelah pembaruan lulus tes, Anda dapat mempromosikan test panggung ke prod panggung. Promosi dapat dilakukan dengan memindahkan API ke prod tahap atau memperbarui nilai variabel tahap dari nama panggung test ke nama. prod

Di bagian ini, kita membahas cara menerapkan API dengan menggunakan [konsol API Gateway](#) atau memanggil API [Gateway REST API](#). [Untuk menggunakan alat lain, lihat dokumentasi AWS CLI atau SDK.AWS](#)

Topik

- [Membuat penerapan untuk REST API di API Gateway](#)

- [Siapkan panggung untuk REST API di API Gateway](#)
- [Siapkan penerapan rilis kenari API Gateway](#)
- [Pembaruan untuk REST APIs yang memerlukan pemindahan](#)

Membuat penerapan untuk REST API di API Gateway

Di API Gateway, penerapan REST API diwakili oleh sumber daya [Deployment](#). Ini mirip dengan executable API yang diwakili oleh sumber daya [RestApi](#)

Agar klien dapat memanggil API Anda, Anda harus membuat penerapan dan mengaitkan tahapan dengannya. Sebuah panggung diwakili oleh sumber daya [Panggung](#). Ini mewakili snapshot API, termasuk metode, integrasi, model, template pemetaan, dan otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus). Saat memperbarui API, Anda dapat menerapkan ulang API dengan mengaitkan tahap baru dengan penerapan yang ada. Kami membahas membuat panggung di [the section called “Siapkan panggung”](#).

Topik

- [Buat deployment](#)
- [Langkah selanjutnya untuk penerapan API](#)

Buat deployment

Prosedur berikut menunjukkan cara membuat penerapan untuk REST API.

Konsol Manajemen AWS

Anda harus telah membuat REST API sebelum menerapkannya untuk pertama kalinya. Untuk informasi selengkapnya, lihat [Kembangkan REST API di API Gateway](#).

Konsol API Gateway memungkinkan Anda men-deploy API dengan membuat deployment dan mengaitkannya dengan tahap baru atau yang sudah ada.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel APInavigasi, pilih API yang ingin Anda terapkan.
3. Di panel Resources, pilih Deploy API.
4. Untuk Stage, pilih dari berikut ini:

- a. Untuk membuat tahap baru, pilih Tahap baru, lalu masukkan nama di Nama panggung. Secara opsional, Anda dapat memberikan deskripsi untuk penyebaran dalam deskripsi Deployment.
 - b. Untuk memilih tahap yang ada, pilih nama panggung dari menu tarik-turun. Anda mungkin ingin memberikan deskripsi penerapan baru dalam deskripsi Deployment.
 - c. Untuk membuat penyebaran yang tidak terkait dengan tahap, pilih Tidak ada tahap. Nanti, Anda dapat mengaitkan penerapan ini dengan sebuah panggung.
5. Pilih Deploy.

AWS CLI

Saat membuat penerapan, Anda membuat instance sumber daya Deployment.

Perintah [create-deployment](#) berikut membuat penerapan baru:

```
aws apigateway create-deployment --rest-api-id rest-api-id
```

Anda tidak dapat memanggil API sampai Anda mengaitkan penerapan ini dengan sebuah panggung. Dengan tahap yang ada, Anda dapat melakukan ini dengan memperbarui properti [deploymentID stage dengan ID](#) deployment yang baru dibuat. Perintah [update-stage](#) berikut memperbarui tahap dengan penerapan baru. Di konsol, ini disebut penyebaran aktif.

```
aws apigateway update-stage \  
  --rest-api-id rest-api-id \  
  --stage-name 'stage-name' \  
  --patch-operations op='replace',path='/deploymentId',value='deployment-id'
```

Saat Anda membuat penerapan, Anda juga dapat mengaitkannya dengan tahap baru secara bersamaan. Perintah [create-deployment](#) berikut membuat penerapan baru dan mengaitkannya dengan tahap baru yang disebut: beta

```
aws apigateway create-deployment \  
  --rest-api-id rest-api-id \  
  --stage-name beta
```

Untuk menerapkan ulang API, lakukan langkah yang sama. Anda dapat menggunakan kembali tahap yang sama.

Langkah selanjutnya untuk penerapan API

Berikut ini adalah langkah selanjutnya untuk penerapan API Anda.

Ubah pengaturan panggung

Setelah API diterapkan, Anda dapat memodifikasi pengaturan panggung untuk mengaktifkan atau menonaktifkan cache API, logging, atau meminta pembatasan. Anda juga dapat memilih sertifikat klien untuk backend untuk mengautentikasi API Gateway dan mengatur variabel tahap untuk meneruskan konteks penerapan ke integrasi API saat runtime. Untuk informasi selengkapnya, silakan lihat [Ubah pengaturan panggung](#)

Setelah memodifikasi pengaturan tahap, Anda harus menerapkan ulang API agar perubahan diterapkan.

Note

Jika pengaturan yang diperbarui, seperti mengaktifkan logging, memerlukan peran IAM baru, Anda dapat menambahkan peran IAM yang diperlukan tanpa menerapkan ulang API. Namun, mungkin perlu beberapa menit sebelum peran IAM baru berlaku. Sebelum itu terjadi, jejak panggilan API Anda tidak dicatat meskipun Anda telah mengaktifkan opsi logging.

Pilih kombinasi tahap penerapan yang berbeda

Karena penerapan merepresentasikan snapshot API dan tahap menentukan jalur ke dalam snapshot, Anda dapat memilih kombinasi tahap penerapan yang berbeda untuk mengontrol cara pengguna memanggil ke versi API yang berbeda. Ini berguna, misalnya, ketika Anda ingin memutar kembali status API ke penerapan sebelumnya atau menggabungkan 'cabang pribadi' API ke publik.

Prosedur berikut menunjukkan cara melakukannya menggunakan Stage Editor di konsol API Gateway. Diasumsikan bahwa Anda harus telah menerapkan API lebih dari sekali.

1. Jika Anda belum berada di panel Tahapan, di panel navigasi utama, pilih Tahapan.
2. Pilih tahap yang ingin Anda perbarui.
3. Pada tab Riwayat Deployment, pilih penyebaran yang ingin Anda gunakan untuk tahap.
4. Pilih Ubah penerapan aktif.

5. Konfirmasikan bahwa Anda ingin mengubah penerapan aktif dan pilih Ubah penerapan aktif di kotak dialog Make active deployment.

Teruskan data khusus penerapan ke API Anda.

Untuk penerapan, Anda dapat menyetel atau memodifikasi variabel tahap untuk meneruskan data khusus penerapan ke integrasi API saat runtime. Anda dapat melakukan ini pada tab Stage Variables di Stage Editor. Untuk informasi lebih lanjut, lihat petunjuk di [Menggunakan variabel stage untuk REST API di API Gateway](#).

Siapkan panggung untuk REST API di API Gateway

Tahap adalah referensi bernama untuk penerapan, yang merupakan snapshot dari API. Anda menggunakan [Stage](#) untuk mengelola dan mengoptimalkan penerapan tertentu. Misalnya, Anda dapat mengonfigurasi pengaturan tahap untuk mengaktifkan caching, menyesuaikan pembatasan permintaan, mengonfigurasi logging, menentukan variabel tahap, atau melampirkan rilis kenari untuk pengujian. Bagian berikut menunjukkan cara membuat dan mengkonfigurasi panggung Anda.

Buat panggung baru

Setelah penerapan awal, Anda dapat menambahkan lebih banyak tahapan dan mengaitkannya dengan penerapan yang ada. Anda dapat menggunakan konsol API Gateway untuk membuat tahap baru, atau Anda dapat memilih tahap yang ada saat menerapkan API. Secara umum, Anda dapat menambahkan tahap baru ke penerapan API sebelum menerapkan ulang API. Untuk membuat tahap baru menggunakan konsol API Gateway, ikuti langkah-langkah berikut:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan di bawah API.
4. Dari panel navigasi Tahapan, pilih Buat tahap.
5. Untuk nama Panggung, masukkan nama, misalnya, **prod**.

Note

Nama panggung hanya dapat berisi karakter alfanumerik, tanda hubung, dan garis bawah. Panjang maksimum adalah 128 karakter.

6. (Opsional). Untuk Deskripsi, masukkan deskripsi panggung.
7. Untuk Deployment, pilih tanggal dan waktu penerapan API yang ada yang ingin Anda kaitkan dengan tahap ini.
8. Di bawah Pengaturan tambahan, Anda dapat menentukan pengaturan tambahan untuk panggung Anda.
9. Pilih Buat panggung.

Ubah pengaturan panggung

Setelah penerapan API berhasil, tahap diisi dengan pengaturan default. Anda dapat menggunakan konsol atau API Gateway REST API untuk mengubah pengaturan panggung, termasuk caching API dan logging. Jika Anda telah memodifikasi titik akhir default REST API Anda, saat Anda memperbarui tahapan, modifikasi akan disebarkan ke semua tahapan API Anda. Langkah-langkah berikut menunjukkan cara melakukannya menggunakan editor Stage dari konsol API Gateway.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan di bawah API.
4. Di panel Tahapan, pilih nama panggung.
5. Di bagian Detail tahap, pilih Edit.
6. (Opsional) Untuk deskripsi Tahap, edit deskripsi.
7. Untuk pengaturan tambahan, Anda mengubah pengaturan berikut:

Pengaturan cache

Untuk mengaktifkan caching API untuk stage, aktifkan cache API Provision. Kemudian konfigurasikan caching tingkat metode Default, Kapasitas cache, Enkripsi data cache, Cache time-to-live (TTL), serta persyaratan apa pun untuk pembatalan cache per kunci.

Caching tidak aktif sampai Anda mengaktifkan caching tingkat metode default atau mengaktifkan cache tingkat metode untuk metode tertentu.

Untuk informasi selengkapnya tentang pengaturan cache, lihat [Pengaturan cache untuk REST APIs di API Gateway](#).

 Note

Jika Anda mengaktifkan caching API untuk tahap API, AWS akun Anda mungkin dikenakan biaya untuk caching API. Caching tidak memenuhi syarat untuk Tingkat AWS Gratis.

Pengaturan pelambatan

Untuk menetapkan target pelambatan tingkat tahap untuk semua metode yang terkait dengan API ini, aktifkan Throttling.

Untuk Rate, masukkan target rate. Ini adalah tarif, dalam permintaan per detik, token ditambahkan ke ember token. Tingkat tingkat tahap tidak boleh lebih dari tingkat [tingkat akun seperti yang ditentukan](#) dalam. [Kuota untuk mengonfigurasi dan menjalankan REST API di API Gateway](#)

Untuk Burst, masukkan target burst rate. Laju burst, adalah kapasitas ember token. Ini memungkinkan lebih banyak permintaan melalui untuk jangka waktu tertentu daripada tingkat target. Tingkat ledakan tingkat tahap ini tidak boleh lebih dari tingkat burst rate [akun seperti yang ditentukan](#) dalam. [Kuota untuk mengonfigurasi dan menjalankan REST API di API Gateway](#)

 Note

Tarif pelambatan bukanlah batas yang sulit, dan diterapkan atas dasar upaya terbaik. Dalam beberapa kasus, klien dapat melebihi target yang Anda tetapkan. Jangan mengandalkan pembatasan untuk mengontrol biaya atau memblokir akses ke API. Pertimbangkan [AWS Budgets](#) untuk menggunakan untuk memantau biaya dan [AWS WAF](#) mengelola permintaan API.

Pengaturan firewall dan sertifikat

Untuk mengaitkan ACL AWS WAF web dengan panggung, pilih ACL web dari daftar dropdown Web ACL. Jika diinginkan, pilih Blokir Permintaan API jika WebACL tidak dapat dievaluasi (Gagal- Tutup).

Untuk memilih sertifikat klien untuk tahap Anda, pilih sertifikat dari menu tarik-turun sertifikat Klien.

8. Pilih Lanjutkan.
9. Tinjau perubahan Anda dan pilih Simpan perubahan.
10. Untuk mengaktifkan CloudWatch Log Amazon untuk semua metode yang terkait dengan tahap API Gateway API ini, di bagian Log dan penelusuran, pilih Edit.

 Note

Untuk mengaktifkan CloudWatch Log, Anda juga harus menentukan ARN dari peran IAM yang memungkinkan API Gateway untuk menulis informasi ke CloudWatch Log atas nama pengguna Anda. Untuk melakukannya, pilih Pengaturan dari panel navigasi API utama. Kemudian, untuk peran CloudWatch log, masukkan ARN dari peran IAM. Untuk skenario aplikasi umum, peran IAM dapat melampirkan kebijakan terkelola [Amazon APIGateway PushToCloudWatchLogs](#).

Peran IAM juga harus berisi pernyataan hubungan kepercayaan berikut:

JSON

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

Untuk informasi selengkapnya CloudWatch, lihat [Panduan CloudWatch Pengguna Amazon](#).

11. Pilih level logging dari menu dropdown CloudWatch Log. Level logging adalah sebagai berikut:

- Mati - Logging tidak diaktifkan untuk tahap ini.
 - Hanya kesalahan - Logging diaktifkan hanya untuk kesalahan.
 - Kesalahan dan info log - Logging diaktifkan untuk semua acara.
12. Pilih Penelusuran data agar API Gateway melaporkan ke CloudWatch pencatatan jejak data untuk tahap Anda. Ini dapat berguna untuk memecahkan masalah APIs, tetapi dapat mengakibatkan pencatatan data sensitif.

 Note

Kami menyarankan Anda untuk tidak menggunakan data tracing untuk produksi APIs.

13. Pilih Metrik terperinci agar API Gateway melaporkan ke CloudWatch metrik API `API calls`, `Latency`, `Integration latency`, `400 errors`, dan `500 errors`. Untuk informasi selengkapnya CloudWatch, lihat [Pemantauan dasar dan pemantauan terperinci](#) di Panduan CloudWatch Pengguna Amazon.

 Important

Akun Anda dikenakan biaya untuk mengakses metrik tingkat metode, tetapi bukan CloudWatch metrik tingkat API atau tingkat tahap.

14. Untuk mengaktifkan pencatatan akses ke tujuan, aktifkan Pencatatan akses khusus.
15. Untuk ARN tujuan log Access, masukkan ARN grup log atau aliran Firehose.

Format ARN untuk Firehose adalah `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. Nama aliran Firehose Anda harus `amazon-apigateway-{your-stream-name}`.

16. Dalam format Log, masukkan format log. Untuk mempelajari lebih lanjut tentang contoh format log, lihat [the section called "CloudWatch format log untuk API Gateway"](#).
17. Untuk mengaktifkan [AWS X-Ray](#) penelusuran untuk tahap API, pilih penelusuran X-Ray. Untuk informasi selengkapnya, lihat [Lacak permintaan pengguna ke REST APIs menggunakan X-Ray di API Gateway](#).
18. Pilih Simpan perubahan. Menerapkan ulang API Anda agar pengaturan baru diterapkan.

Ganti pengaturan tingkat panggung

Setelah menyesuaikan setelan tingkat tahap, Anda dapat menggantinya untuk setiap metode API. Beberapa opsi ini dapat mengakibatkan biaya tambahan untuk Anda Akun AWS.

1. Untuk mengonfigurasi penggantian metode, perluas panggung di bawah panel navigasi sekunder, lalu pilih metode.

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. Untuk penggantian Metode, pilih Edit.
3. Untuk mengaktifkan CloudWatch pengaturan tingkat metode, untuk CloudWatch Log, pilih level logging.
4. Untuk mengaktifkan pencatatan jejak data untuk metode Anda, pilih Penelusuran data.

 Note

Kami menyarankan Anda untuk tidak menggunakan data tracing untuk produksi APIs.

5. Untuk mengaktifkan metrik detail tingkat metode, pilih Metrik terperinci. Akun Anda dikenakan biaya untuk mengakses metrik tingkat metode, tetapi bukan CloudWatch metrik tingkat API atau tingkat tahap.
6. Untuk mengaktifkan pelambatan tingkat metode, pilih Throttling. Masukkan opsi tingkat metode yang sesuai. Untuk mempelajari selengkapnya tentang throttling, lihat [the section called “Throttling”](#)
7. Untuk mengkonfigurasi cache tingkat metode, pilih Aktifkan cache metode. Jika Anda mengubah pengaturan caching tingkat metode default di detail Tahap, itu tidak memengaruhi pengaturan ini.
8. Pilih Simpan.

Tambahkan API REST API Gateway sebagai target untuk Amazon Bedrock AgentCore Gateway

Amazon Bedrock AgentCore Gateway memberi pengembang agen AI cara aman untuk mengekspos REST API Gateway Anda APIs sebagai alat yang kompatibel dengan Model Context Protocol (MCP). AgentCore Gateway menggunakan target untuk menentukan alat. Saat Anda menambahkan tahap sebagai target, Gateway Anda menjadi URL MCP tunggal yang memungkinkan akses ke alat untuk agen. Untuk informasi selengkapnya, lihat [Tahapan API Gateway REST API sebagai target](#) di Panduan Pengembang Amazon Bedrock AgentCore Gateway.

Target API Gateway menghubungkan AgentCore Gateway Anda ke tahapan REST Anda APIs. Anda dapat memasukkan seluruh tahap sebagai target, atau memilih sumber daya. Setelah Anda membuat target API Gateway, AgentCore Gateway menerjemahkan permintaan MCP yang masuk ke dalam permintaan HTTP dan menangani pemformatan respons. Klien MCP dapat mengambil dokumentasi API menggunakan `tools/list` metode dan memanggil APIs menggunakan metode. `tools/call`

Pertimbangan-pertimbangan

Pertimbangan berikut mungkin memengaruhi penggunaan Anda menambahkan tahap sebagai target ke AgentCore Gateway:

- Anda harus sudah memiliki AgentCore Gateway.
- Hanya REST publik APIs yang didukung.
- Titik akhir default API Anda tidak dapat dinonaktifkan.

- Setiap metode API Anda harus memiliki [nama operasi](#) yang ditentukan untuknya, atau kebutuhan Anda untuk membuat penggantian nama saat Anda menambahkan tahap sebagai target. Nama ini digunakan sebagai nama alat yang digunakan agen untuk berinteraksi dengan metode Anda.
- Anda dapat menggunakan `API_KEY,NO_AUTH`, atau jenis penyedia `GATEWAY_IAM_ROLE` kredensi untuk Outbound Auth agar Gateway Anda dapat mengakses API Anda. Penyedia `API_KEY` kredensi ditentukan oleh AgentCore Gateway. Anda dapat menggunakan kunci API Gateway API yang ada. Untuk informasi selengkapnya, lihat [Menyiapkan Autentikasi Keluar](#).
- Jika Anda menggunakan kumpulan pengguna Amazon Cognito atau otorisasi Lambda untuk mengontrol akses ke API Anda, klien MCP tidak dapat mengaksesnya.
- API Anda harus berada di akun dan Wilayah yang sama dengan AgentCore Gateway Anda.

Menambahkan tahap API sebagai target untuk AgentCore Gateway

Prosedur berikut menunjukkan cara menambahkan tahap API sebagai target untuk AgentCore Gateway.

Untuk menambahkan tahap API sebagai target untuk AgentCore Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API yang diterapkan ke panggung.
3. Di panel navigasi utama, pilih Tahapan.
4. Pilih Tindakan tahap, lalu pilih Buat target MCP.
5. Untuk AgentCore Gateway, pilih AgentCore Gateway.
6. Untuk nama Target, masukkan nama target.
7. Untuk deskripsi Target, masukkan deskripsi.
8. Simpan API dan panggung yang disediakan.
9. Untuk sumber daya API Pilih, pilih sumber daya API Anda yang dapat diakses oleh agen yang menggunakan AgentCore Gateway Anda.

Jika Anda tidak memilih sumber daya, agen tidak dapat melihat dokumentasi atau memanggil titik akhir.

10. Kombinasi sumber daya dan metode adalah operasi untuk alat. Jika operasi Anda tidak memiliki nama, buat penggantian nama.

Anda juga dapat menentukan nama operasi untuk metode saat Anda membuatnya.

11. Untuk konfigurasi Outbound Auth, pilih Peran IAM, Tanpa otorisasi, atau kunci API.
12. Pilih Buat target.

Untuk melihat semua AgentCore Gateway yang memiliki akses ke Anda APIs, pilih bagian target MCP di panel navigasi utama. Di bagian ini, Anda dapat membuat target MCP untuk API apa pun di Wilayah yang diterapkan ke sebuah panggung. Pilih Buat target MCP dan ikuti langkah-langkah sebelumnya.

Anda juga dapat melihat alat yang tersedia untuk target Anda dan mengedit target Anda di konsol AgentCore Gateway. Untuk informasi selengkapnya, lihat [Menambahkan target ke AgentCore Gateway yang ada](#).

Hapus panggung

Ketika Anda tidak lagi membutuhkan panggung, Anda dapat menghapusnya untuk menghindari membayar sumber daya yang tidak digunakan. Langkah-langkah berikut menunjukkan kepada Anda cara menggunakan konsol API Gateway untuk menghapus panggung.

Warning

Menghapus tahap dapat menyebabkan sebagian atau semua API terkait tidak dapat digunakan oleh pemanggil API. Menghapus tahap tidak dapat dibatalkan, tetapi Anda dapat membuat ulang panggung dan mengaitkannya dengan penerapan yang sama.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan.
4. Di panel Tahapan, pilih tahap yang ingin Anda hapus, lalu pilih Tindakan tahap, Hapus tahap.
5. Saat Anda diminta, masukkan **confirm**, lalu pilih Hapus.

Menyiapkan tag untuk tahap API di API Gateway

Di API Gateway, Anda dapat menambahkan tag ke tahap API, menghapus tag dari tahap, atau melihat tag. Untuk melakukannya, Anda dapat menggunakan konsol API Gateway, AWS CLI/SDK, atau API Gateway REST API.

Sebuah panggung juga dapat mewarisi tag dari REST API induknya. Untuk informasi selengkapnya, lihat [the section called “Warisan tag di Amazon API Gateway V1”](#).

Untuk informasi selengkapnya tentang menandai resource API Gateway, lihat [Penandaan](#).

Topik

- [Menyiapkan tag untuk tahap API menggunakan konsol API Gateway](#)
- [Menyiapkan tag untuk tahap API menggunakan AWS CLI](#)
- [Menyiapkan tag untuk tahap API menggunakan API Gateway REST API](#)

Menyiapkan tag untuk tahap API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara menyiapkan tag untuk tahap API.

Untuk menyiapkan tag untuk tahap API dengan menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway.
2. Pilih API yang sudah ada, atau buat API baru yang menyertakan sumber daya, metode, dan integrasi terkait.
3. Pilih tahapan atau terapkan API ke tahap baru.
4. Di panel navigasi utama, pilih Tahapan.
5. Pilih tab Tanda. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
6. Pilih Kelola tanda.
7. Di Editor Tag, pilih Tambahkan tag. Masukkan kunci tag (misalnya, `Department`) di bidang Kunci, dan masukkan nilai tag (misalnya, `Sales`) di bidang Nilai. Pilih Simpan untuk menyimpan tag.
8. Jika perlu, ulangi langkah 5 untuk menambahkan lebih banyak tag ke tahap API. Jumlah maksimum tag per tahap adalah 50.
9. Untuk menghapus tag yang ada dari panggung, pilih Hapus.
10. Jika API telah digunakan sebelumnya di konsol API Gateway, Anda perlu menerapkannya kembali agar perubahan diterapkan.

Menyiapkan tag untuk tahap API menggunakan AWS CLI

Anda dapat mengatur tag untuk tahap API menggunakan perintah [create-stage](#) atau perintah [tag-resource](#). AWS CLI Anda dapat menghapus satu atau beberapa tag dari tahap API menggunakan perintah [untag-resource](#).

Perintah [create-stage](#) berikut menambahkan tag saat membuat panggung: test

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

Perintah [tag-resource](#) berikut menambahkan tag ke panggung: prod

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/restapis/abc123/stages/prod --tags Department=Sales
```

Perintah [untag-resource](#) berikut menghapus Department=Sales tag dari panggung: test

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/restapis/abc123/stages/test --tag-keys Department
```

Menyiapkan tag untuk tahap API menggunakan API Gateway REST API

Anda dapat menyiapkan tag untuk tahap API menggunakan API Gateway REST API dengan melakukan salah satu hal berikut:

- Panggilan [tags:tag](#) untuk menandai tahap API.
- Panggilan [tags:untag](#) untuk menghapus satu atau beberapa tag dari tahap API.
- Panggil [stage:create](#) untuk menambahkan satu atau beberapa tag ke tahap API yang Anda buat.

Anda juga dapat menelepon [tags:get](#) untuk mendeskripsikan tag dalam tahap API.

Menandai tahap API

Setelah Anda menerapkan API (m5zr3vnks7) ke stage (test), beri tag stage dengan memanggil [tags:tag](#). Tahap yang diperlukan Nama Sumber Daya Amazon (ARN) (arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test) harus dikodekan URL (). 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"
  }
}
```

Anda juga dapat menggunakan permintaan sebelumnya untuk memperbarui tag yang ada ke nilai baru.

Anda dapat menambahkan tag ke panggung saat memanggil [stage:create](#) untuk membuat panggung:

```
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"
  }
}
```

Membatalkan tag tahap API

Untuk menghapus Department tag dari panggung, hubungi [tags:untag](#):

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Untuk menghapus lebih dari satu tag, gunakan daftar kunci tag yang dipisahkan koma dalam ekspresi kueri—misalnya, `?tagKeys=Department,Division,...`

Jelaskan tag untuk tahap API

Untuk mendeskripsikan tag yang ada pada tahap tertentu, panggil [tags:get](#):

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Respons yang berhasil mirip dengan yang berikut:

```
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"
  }
}
```

Menggunakan variabel stage untuk REST API di API Gateway

Variabel tahap adalah pasangan nilai kunci yang dapat Anda definisikan sebagai atribut konfigurasi yang terkait dengan tahap penerapan REST API. Mereka bertindak seperti variabel lingkungan dan

dapat digunakan dalam template penyiapan dan pemetaan API Anda. Dengan tahapan penerapan di API Gateway, Anda dapat mengelola beberapa tahapan rilis untuk setiap API dan menggunakan variabel tahap, Anda dapat mengonfigurasi tahap penerapan API untuk berinteraksi dengan titik akhir backend yang berbeda.

Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensial. Untuk meneruskan data sensitif ke integrasi, gunakan AWS Lambda otorisasi. Anda dapat meneruskan data sensitif ke integrasi dalam output otorisasi Lambda. Untuk mempelajari selengkapnya, lihat [the section called “Keluaran dari otorisasi API Gateway Lambda”](#).

Gunakan kasus untuk variabel tahap

Berikut ini adalah kasus penggunaan untuk variabel tahap Anda.

Tentukan titik akhir backend yang berbeda

API Anda dapat meneruskan GET permintaan sebagai proxy HTTP ke host web backend. Anda dapat menggunakan variabel stage sehingga ketika pemanggil API memanggil titik akhir produksi Anda, API Gateway akan memanggil `example.com`. Kemudian, ketika pemanggil API memanggil tahap beta, API Gateway memanggil host web yang berbeda, seperti `beta.example.com`. Demikian pula, variabel tahap dapat digunakan untuk menentukan nama AWS Lambda fungsi yang berbeda untuk setiap tahap di API Anda. Anda tidak dapat menggunakan variabel tahap untuk menetapkan titik akhir integrasi yang berbeda, seperti mengarahkan GET permintaan ke integrasi proxy HTTP dalam satu tahap dan integrasi proxy Lambda di tahap lain.

Saat menentukan nama fungsi Lambda sebagai nilai variabel tahap, Anda harus mengonfigurasi izin pada fungsi Lambda secara manual. Saat Anda menentukan fungsi Lambda di konsol API Gateway, sebuah AWS CLI perintah akan muncul untuk mengonfigurasi izin yang tepat. Anda juga dapat menggunakan AWS CLI perintah berikut untuk melakukan ini.

```
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
```

Lulus informasi menggunakan templat pemetaan

Anda dapat mengakses variabel tahap dalam template pemetaan, atau meneruskan parameter konfigurasi ke backend AWS Lambda atau HTTP Anda. Misalnya, Anda mungkin ingin menggunakan kembali fungsi Lambda yang sama untuk beberapa tahap di API Anda, tetapi

fungsi tersebut harus membaca data dari tabel Amazon DynamoDB yang berbeda tergantung pada tahapannya. Dalam template pemetaan yang menghasilkan permintaan untuk fungsi Lambda, Anda dapat menggunakan variabel tahap untuk meneruskan nama tabel ke Lambda.

Untuk menggunakan variabel tahap, pertama-tama Anda mengkonfigurasi variabel tahap, dan kemudian Anda menetapkan nilai. Misalnya, untuk menyesuaikan titik akhir integrasi HTTP, pertama buat variabel `url stage`, lalu di permintaan integrasi API Anda, masukkan nilai variabel `stage`, `http://{stageVariables.url}`. Nilai ini memberi tahu API Gateway untuk mengganti variabel stage Anda `{}` saat runtime, tergantung pada tahap mana API Anda berjalan. Untuk informasi selengkapnya, lihat [the section called “Siapkan variabel tahap untuk REST APIs di API Gateway”](#).

Siapkan variabel tahap untuk REST APIs di API Gateway

Bagian ini menunjukkan cara menyiapkan berbagai variabel tahap untuk dua tahapan penerapan API sampel dengan menggunakan konsol Amazon API Gateway. Untuk memahami cara menggunakan variabel tahap di API Gateway, kami sarankan Anda mengikuti semua prosedur di bagian ini.

Prasyarat

Sebelum Anda mulai, pastikan prasyarat berikut terpenuhi:

- Anda harus memiliki API yang tersedia di API Gateway. Ikuti petunjuk dalam [Kembangkan REST API di API Gateway](#).
- Anda harus telah menerapkan API setidaknya sekali. Ikuti petunjuk dalam [Menerapkan REST APIs di API Gateway](#).
- Anda harus telah membuat tahap pertama untuk API yang diterapkan. Ikuti petunjuk dalam [Buat panggung baru](#).

Memanggil titik akhir HTTP melalui API dengan variabel tahap

Prosedur ini menjelaskan cara membuat variabel tahap untuk titik akhir HTTP dan dua tahap untuk API Anda. Selain itu, Anda membuat variabel tahap, `url`, `stageName`, dan `function` yang digunakan dalam prosedur berikut di bagian ini.

Untuk memanggil titik akhir HTTP melalui API dengan variabel tahap

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Buat API, lalu buat GET metode pada sumber daya root API. Atur tipe integrasi ke HTTP dan atur URL Endpoint ke **`http://${stageVariables.url}`**.
3. Menerapkan API ke tahap baru bernama **beta**.
4. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap beta.
5. Pada tab variabel Stage, pilih Edit.
6. Pilih Tambahkan variabel tahap.
7. Untuk Nama, masukkan **url**. Untuk nilai, masukkan **`httpbin.org/get`**.
8. Pilih Tambahkan variabel tahap, dan kemudian lakukan hal berikut:

Untuk Nama, masukkan **stageName**. Untuk nilai, masukkan **beta**.
9. Pilih Tambahkan variabel tahap, dan kemudian lakukan hal berikut:

Untuk Nama, masukkan **function**. Untuk nilai, masukkan **HelloWorld**.
10. Pilih Simpan.
11. Sekarang buat tahap kedua. Dari panel navigasi Tahapan, pilih Buat tahap. Untuk nama Panggung, masukkan **prod**. Pilih penerapan terbaru dari Deployment, lalu pilih Create stage.
12. Seperti tahap beta, atur variabel tiga tahap yang sama (url, stageName, dan fungsi) ke nilai yang berbeda (**`petstore-demo-endpoint.execute-api.com/petstore/pets`**, dan **HelloEveryone**)**prod**, masing-masing.
13. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Ini memulai GET permintaan tahap beta pada sumber daya root API.

 Note

Tautan URL Invoke menunjuk ke sumber daya root API dalam tahap beta. Memasukkan URL di browser web memanggil GET metode tahap beta pada sumber daya root. Jika metode didefinisikan pada sumber daya anak dan bukan pada sumber daya root itu sendiri, memasukkan URL di browser web mengembalikan respons `{"message": "Missing Authentication Token"}` kesalahan. Dalam hal ini, Anda harus menambahkan nama sumber daya anak tertentu ke tautan URL Panggilan.

14. Respons yang Anda dapatkan dari GET permintaan tahap beta ditampilkan berikutnya. Anda juga dapat memverifikasi hasilnya dengan menggunakan browser untuk menavigasi ke `http://`

httpbin.org/get. Nilai ini ditetapkan ke `url` variabel dalam tahap beta. Kedua tanggapan itu identik.

15. Di panel navigasi Tahapan, pilih tahap prod. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Ini memulai GET permintaan tahap prod pada sumber daya root API.
16. Respons yang Anda dapatkan dari GET permintaan tahap prod ditampilkan berikutnya. Anda dapat memverifikasi hasilnya dengan menggunakan browser untuk menavigasi ke `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Nilai ini ditetapkan ke `url` variabel dalam tahap prod. Kedua tanggapan itu identik.

Lulus metadata khusus tahap ke backend HTTP

Prosedur ini menjelaskan cara menggunakan nilai variabel tahap dalam ekspresi parameter kueri untuk meneruskan metadata spesifik tahap ke backend HTTP. Kami akan menggunakan variabel `stageName` tahap yang dinyatakan dalam prosedur sebelumnya.

Untuk meneruskan metadata khusus tahap ke backend HTTP

1. Di panel navigasi Resource, pilih metode GET.

Untuk menambahkan parameter string kueri ke URL metode, pilih tab Permintaan metode, lalu di bagian Pengaturan permintaan metode, pilih Edit.

2. Pilih parameter string kueri URL dan lakukan hal berikut:
 - a. Pilih Tambahkan string kueri.
 - b. Untuk Nama, masukkan **`stageName`**.
 - c. Tetap Diperlukan dan Caching dimatikan.
3. Pilih Simpan.
4. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan permintaan integrasi, pilih Edit.
5. Untuk URL Endpoint, tambahkan **`?stageName=${stageVariables.stageName}`** ke nilai URL yang ditentukan sebelumnya, sehingga seluruh URL Endpoint adalah **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**
6. Pilih Deploy API dan pilih tahap beta.

7. Di panel navigasi utama, pilih Tahapan. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web.

 Note

Kami menggunakan tahap beta di sini karena titik akhir HTTP (seperti yang ditentukan oleh `url` variabel `"http://httpbin.org/get "`) menerima ekspresi parameter kueri dan mengembalikannya sebagai `args` objek dalam responsnya.

8. Anda mendapatkan tanggapan berikut. Perhatikan bahwa `beta`, ditugaskan ke variabel `stageName` tahap, diteruskan di backend sebagai argumen. `stageName`

```
{
  "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-1111111111111111;Root=1-11111111-1111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Memanggil fungsi Lambda melalui API dengan variabel panggung

Prosedur ini menjelaskan cara menggunakan variabel stage untuk memanggil fungsi Lambda sebagai backend API Anda. Anda menggunakan variabel `function` tahap yang dideklarasikan dalam [Memanggil titik akhir HTTP melalui API dengan variabel tahap](#).

Saat menyetel fungsi Lambda sebagai nilai variabel tahap, gunakan nama lokal fungsi, mungkin termasuk alias atau spesifikasi versinya, seperti dalam **HelloWorld**, atau. **HelloWorld:1** **HelloWorld:alpha** Jangan gunakan ARN fungsi (misalnya, **arn:aws:lambda:us-east-1:123456789012:function:HelloWorld**). Konsol API Gateway mengasumsikan nilai variabel stage untuk fungsi Lambda sebagai nama fungsi yang tidak memenuhi syarat dan memperluas variabel tahap yang diberikan menjadi ARN.

Untuk menjalankan fungsi Lambda melalui API dengan variabel stage

1. Buat fungsi Lambda bernama **HelloWorld** menggunakan runtime Node.js default. Kode harus berisi yang berikut:

```
export const handler = function(event, context, callback) {  
    if (event.stageName)  
        callback(null, 'Hello, World! I\'m calling from the ' + event.stageName + '  
stage.');
```

Untuk informasi selengkapnya tentang cara membuat fungsi Lambda, lihat [Memulai konsol REST API](#).

2. Di panel Resources, pilih Buat sumber daya, lalu lakukan hal berikut:
 - a. Untuk jalur Sumber Daya, pilih/.
 - b. Untuk Nama sumber daya, masukkan **lambdav1**.
 - c. Pilih Buat sumber daya.
3. Pilih sumber daya /lambdav1, lalu pilih Create method.

Kemudian, lakukan hal berikut:

- a. Untuk tipe Metode, pilih GET.
- b. Untuk jenis Integrasi, pilih fungsi Lambda.
- c. Matikan integrasi proxy Lambda.
- d. Untuk fungsi Lambda, masukkan. `${stageVariables.function}`

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▼

🔍

`${stageVariables.function}`

✕

Tip

Saat diminta dengan perintah Tambah izin, salin perintah [add-permission](#). Jalankan perintah pada setiap fungsi Lambda yang akan ditugaskan ke variabel function

panggung. Misalnya, jika `$stageVariables.function` nilainya `HelloWorld`, jalankan AWS CLI perintah berikut:

```
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
```

Gagal melakukannya menghasilkan `500 Internal Server Error` respons saat menjalankan metode. Ganti `${stageVariables.function}` dengan nama fungsi Lambda yang ditetapkan ke variabel `stage`.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▼

Q `${stageVariables.function}` X

 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/lambdav1" \
4 --principal apigateway.amazonaws.com \
5 --statement-id abcd-12345-efg \
6 --action lambda:InvokeFunction
```

- e. Pilih metode `Buat`.
4. Terapkan API ke tahap `prod` dan `beta`.
5. Di panel navigasi utama, pilih `Tahapan`. Di panel navigasi `Tahapan`, pilih `beta`. Di bawah `Detail tahap`, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Tambahkan `/lambdav1` ke URL sebelum Anda menekan enter.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Lewati metadata spesifik tahap ke fungsi Lambda melalui variabel panggung

Prosedur ini menjelaskan cara menggunakan variabel tahap untuk meneruskan metadata konfigurasi spesifik tahap ke dalam fungsi Lambda. Anda membuat POST metode dan template pemetaan masukan untuk menghasilkan payload menggunakan variabel `stageName` stage yang Anda deklarasikan sebelumnya.

Untuk meneruskan metadata spesifik tahap ke fungsi Lambda melalui variabel tahap

1. Pilih sumber daya `/lambdav1`, lalu pilih **Create method**.

Kemudian, lakukan hal berikut:

- a. Untuk jenis Metode, pilih **POST**.
 - b. Untuk jenis Integrasi, pilih **fungsi Lambda**.
 - c. Matikan integrasi proxy Lambda.
 - d. Untuk fungsi Lambda, masukkan `${stageVariables.function}`
 - e. Saat diminta dengan perintah Tambah izin, salin perintah [add-permission](#). Jalankan perintah pada setiap fungsi Lambda yang akan ditugaskan ke variabel `function` panggung.
 - f. Pilih metode **Buat**.
2. Pilih tab **Permintaan integrasi**, dan kemudian di bagian **Pengaturan permintaan integrasi**, pilih **Edit**.
 3. Pilih **Templat pemetaan**, lalu pilih **Tambahkan templat pemetaan**.
 4. Untuk jenis Konten, masukkan **`application/json`**.
 5. Untuk badan Template, masukkan template berikut:

```
#set($inputRoot = $input.path('$'))
{
  "stageName" : "$stageVariables.stageName"
}
```

Note

Dalam template pemetaan, variabel tahap harus direferensikan dalam tanda kutip (seperti dalam `"$stageVariables.stageName"`)

atau"`${stageVariables.stageName}`"). Di tempat lain, itu harus direferensikan tanpa tanda kutip (seperti dalam"`${stageVariables.function}`").

6. Pilih Simpan.
7. Terapkan API ke tahap beta dan prod.
8. Untuk menggunakan klien REST API untuk meneruskan metadata khusus tahap, lakukan hal berikut:
 - a. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di bidang input klien REST API. Tambahkan **/lambdav1** sebelum Anda mengirimkan permintaan Anda.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the beta stage."
```

- b. Di panel navigasi Tahapan, pilih prod. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di bidang input klien REST API. Tambahkan **/lambdav1** sebelum Anda mengirimkan permintaan Anda.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the prod stage."
```

9. Untuk menggunakan fitur Uji untuk meneruskan metadata khusus tahapan, lakukan hal berikut:
 - a. Di panel navigasi Sumber daya, pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
 - b. Untuk fungsi, masukkan **HelloWorld**.
 - c. Untuk StageName, masukkan. **beta**
 - d. Pilih Uji. Anda tidak perlu menambahkan badan ke POST permintaan Anda.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the beta stage."
```

- e. Anda dapat mengulangi langkah-langkah sebelumnya untuk menguji tahap Prod. Untuk StageName, masukkan. **Prod**

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the prod stage."
```

Referensi variabel tahap API Gateway untuk REST APIs di API Gateway

Anda dapat menggunakan variabel tahap API Gateway dalam kasus berikut.

Ekspresi pemetaan parameter

Variabel tahap dapat digunakan dalam ekspresi pemetaan parameter untuk permintaan metode API atau parameter header respons, tanpa substitusi sebagian. Dalam contoh berikut, variabel tahap direferensikan tanpa \$ dan {...} melampirkan.

- `stageVariables.<variable_name>`

Templat pemetaan

Variabel tahap dapat digunakan di mana saja dalam template pemetaan, seperti yang ditunjukkan pada contoh berikut.

- `{ "name" : "$stageVariables.<variable_name>" }`
- `{ "name" : "${stageVariables.<variable_name>}" }`

Integrasi HTTP URIs

Variabel tahap dapat digunakan sebagai bagian dari URL integrasi HTTP, seperti yang ditunjukkan pada contoh berikut:

- URI lengkap tanpa protokol — `http://${stageVariables.<variable_name>}`
- Domain lengkap — `http://${stageVariables.<variable_name>}/resource/operation`
- Sebuah subdomain — `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Sebuah jalan — `http://example.com/${stageVariables.<variable_name>}/bar`
- Sebuah string kueri - `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS integrasi URIs

Variabel tahap dapat digunakan sebagai bagian dari tindakan AWS URI atau komponen jalur, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS integrasi URIs (fungsi Lambda)

Variabel tahap dapat digunakan sebagai pengganti nama fungsi Lambda, atau versi/alias, seperti yang ditunjukkan pada contoh berikut.

- `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

Untuk menggunakan variabel stage untuk fungsi Lambda, fungsi harus berada di akun yang sama dengan API. Variabel tahap tidak mendukung fungsi Lambda lintas akun.

Kumpulan pengguna Amazon Cognito

Variabel tahap dapat digunakan sebagai pengganti kumpulan pengguna Amazon Cognito untuk otorisasi. COGNITO_USER_POOLS

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWS kredensi integrasi

Variabel tahap dapat digunakan sebagai bagian dari ARN kredensi AWS pengguna/peran, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Siapkan penerapan rilis kenari API Gateway

[Rilis Canary](#) adalah strategi pengembangan perangkat lunak di mana versi baru API (serta perangkat lunak lainnya) digunakan untuk tujuan pengujian, dan versi dasar tetap digunakan sebagai rilis produksi untuk operasi normal pada tahap yang sama. Untuk tujuan diskusi, kami merujuk ke versi dasar sebagai rilis produksi dalam dokumentasi ini. Meskipun ini masuk akal, Anda bebas menerapkan rilis kenari pada versi non-produksi apa pun untuk pengujian.

Dalam penerapan rilis kenari, total lalu lintas API dipisahkan secara acak menjadi rilis produksi dan rilis kenari dengan rasio yang telah dikonfigurasi sebelumnya. Biasanya, rilis kenari menerima sebagian kecil lalu lintas API dan rilis produksi memakan sisanya. Fitur API yang diperbarui hanya dapat dilihat oleh lalu lintas API melalui canary. Anda dapat menyesuaikan persentase lalu lintas kenari untuk mengoptimalkan cakupan atau kinerja pengujian.

Dengan menjaga lalu lintas kenari kecil dan pemilihan acak, sebagian besar pengguna tidak terpengaruh setiap saat oleh potensi bug di versi baru, dan tidak ada satu pengguna pun yang terpengaruh sepanjang waktu.

Setelah metrik pengujian memenuhi persyaratan Anda, Anda dapat mempromosikan rilis kenari ke rilis produksi dan menonaktifkan kenari dari penerapan. Ini membuat fitur-fitur baru tersedia dalam tahap produksi.

Topik

- [Penerapan rilis Canary di API Gateway](#)
- [Buat penerapan rilis kenari](#)
- [Perbarui rilis kenari](#)
- [Promosikan pelepasan kenari](#)
- [Matikan pelepasan kenari](#)

Penerapan rilis Canary di API Gateway

Di API Gateway, penerapan rilis kenari menggunakan tahap penerapan untuk rilis produksi versi dasar API, dan melampirkan ke tahap rilis kenari untuk versi baru, relatif terhadap versi dasar, API. Tahap ini dikaitkan dengan penyebaran awal dan kenari dengan penyebaran berikutnya. Pada awalnya, baik stage maupun canary menunjuk ke versi API yang sama. Kami menggunakan rilis panggung dan produksi secara bergantian dan menggunakan pelepasan kenari dan kenari secara bergantian di seluruh bagian ini.

[Untuk menerapkan API dengan rilis kenari, Anda membuat penerapan rilis kenari dengan menambahkan pengaturan kenari ke tahap penerapan reguler.](#) Pengaturan kenari menjelaskan rilis kenari yang mendasarinya dan tahapannya mewakili rilis produksi API dalam penerapan ini. Untuk menambahkan pengaturan kenari, atur `canarySettings` pada tahap penerapan dan tentukan yang berikut ini:

- ID penerapan, awalnya identik dengan ID penerapan versi dasar yang ditetapkan di panggung.
- [Persentase lalu lintas API](#), antara 0,0 dan 100,0 inklusif, untuk rilis kenari.
- [Variabel tahap untuk rilis kenari](#) yang dapat mengganti variabel tahap rilis produksi.
- [Penggunaan cache panggung](#) untuk permintaan kenari, jika [useStageCache](#) disetel dan caching API diaktifkan di atas panggung.

Setelah rilis kenari diaktifkan, tahap penerapan tidak dapat dikaitkan dengan penerapan rilis non-kenari lainnya hingga rilis kenari dinonaktifkan dan pengaturan kenari dihapus dari panggung.

Saat Anda mengaktifkan logging eksekusi API, rilis canary memiliki log dan metriknya sendiri yang dihasilkan untuk semua permintaan canary. Mereka dilaporkan ke grup CloudWatch log Log tahap produksi serta grup log CloudWatch Log khusus kenari. Hal yang sama berlaku untuk mengakses logging. Log khusus kenari yang terpisah sangat membantu untuk memvalidasi perubahan API baru dan memutuskan apakah akan menerima perubahan dan mempromosikan rilis kenari ke tahap produksi, atau untuk membuang perubahan dan mengembalikan rilis kenari dari tahap produksi.

Grup log eksekusi tahap produksi diberi nama `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` dan grup log eksekusi rilis kenari diberi nama `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Untuk akses logging, Anda harus membuat grup log baru atau memilih yang sudah ada. Nama grup log akses rilis kenari memiliki `/Canary` akhiran yang ditambahkan ke nama grup log yang dipilih.

Rilis kenari dapat menggunakan cache panggung, jika diaktifkan, untuk menyimpan respons dan menggunakan entri yang di-cache untuk mengembalikan hasil ke permintaan kenari berikutnya, dalam periode pra-konfigurasi time-to-live (TTL).

Dalam penerapan rilis kenari, rilis produksi dan rilis canary API dapat dikaitkan dengan versi yang sama atau dengan versi yang berbeda. Ketika mereka dikaitkan dengan versi yang berbeda, respons untuk permintaan produksi dan kenari di-cache secara terpisah dan cache tahap mengembalikan hasil yang sesuai untuk permintaan produksi dan kenari. Ketika rilis produksi dan rilis kenari dikaitkan dengan penerapan yang sama, cache tahap menggunakan kunci cache tunggal untuk kedua jenis

permintaan dan mengembalikan respons yang sama untuk permintaan yang sama dari rilis produksi dan rilis kenari.

Buat penerapan rilis kenari

[Anda membuat penerapan rilis kenari saat menerapkan API dengan pengaturan canary sebagai input tambahan untuk operasi pembuatan penerapan.](#)

Anda juga dapat membuat penerapan rilis kenari dari penerapan non-canary yang ada dengan membuat [stage:update](#) permintaan untuk menambahkan pengaturan kenari di panggung.

Saat membuat penyebaran rilis non-canary, Anda dapat menentukan nama panggung yang tidak ada. API Gateway membuat satu jika tahap yang ditentukan tidak ada. Namun, Anda tidak dapat menentukan nama panggung yang tidak ada saat membuat penerapan rilis kenari. Anda akan mendapatkan kesalahan dan API Gateway tidak akan membuat penerapan rilis kenari apa pun.

Anda dapat membuat penerapan rilis canary di API Gateway menggunakan konsol API Gateway, the AWS CLI, atau SDK. AWS

Topik

- [Buat penerapan canary menggunakan konsol API Gateway](#)
- [Buat penerapan kenari menggunakan AWS CLI](#)

Buat penerapan canary menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk membuat penerapan rilis canary, ikuti petunjuk di bawah ini:

Untuk membuat penerapan rilis kenari awal

1. Masuk ke konsol API Gateway.
2. Pilih REST API yang ada atau buat REST API baru.
3. Di panel navigasi utama, pilih Resources, lalu pilih Deploy API. Ikuti petunjuk di layar di Deploy API untuk menerapkan API ke tahap baru.

Sejauh ini, Anda telah menerapkan API ke tahap rilis produksi. Selanjutnya, Anda mengonfigurasi pengaturan kenari di panggung dan, jika perlu, juga mengaktifkan caching, mengatur variabel tahap, atau mengonfigurasi eksekusi API atau log akses.

4. Untuk mengaktifkan caching API atau mengaitkan ACL AWS WAF web dengan stage, di bagian Detail tahap, pilih Edit. Untuk informasi selengkapnya, lihat [the section called “Pengaturan cache”](#) atau [the section called “Untuk mengaitkan ACL AWS WAF web dengan tahap API Gateway API menggunakan konsol API Gateway”](#).
5. Untuk mengonfigurasi eksekusi atau mengakses logging, di bagian Log dan penelusuran, pilih Edit dan ikuti petunjuk di layar. Untuk informasi selengkapnya, lihat [Siapkan CloudWatch logging untuk REST APIs di API Gateway](#).
6. Untuk mengatur variabel tahap, pilih tab variabel Tahap dan ikuti petunjuk di layar untuk menambah atau memodifikasi variabel tahap. Untuk informasi selengkapnya, lihat [the section called “Gunakan variabel tahap”](#).
7. Pilih tab Canary, lalu pilih Create canary. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Canary.
8. Di bawah pengaturan Canary, untuk Canary, masukkan persentase permintaan yang akan dialihkan ke kenari.
9. Jika diinginkan, pilih Cache tahap untuk mengaktifkan caching untuk rilis kenari. Cache tidak tersedia untuk rilis canary sampai caching API diaktifkan.
10. Untuk mengganti variabel tahap yang ada, untuk penggantian Canary, masukkan nilai variabel tahap baru.

Setelah rilis canary diinisialisasi pada tahap penerapan, Anda mengubah API dan ingin menguji perubahannya. Anda dapat menerapkan ulang API ke tahap yang sama sehingga versi yang diperbarui dan versi dasar dapat diakses melalui tahap yang sama. Langkah-langkah berikut menjelaskan cara melakukannya.

Untuk menerapkan versi API terbaru ke kenari

1. Dengan setiap pembaruan API, pilih Deploy API.
2. Di Deploy API, pilih tahap yang berisi kenari dari daftar dropdown tahap Deployment.
3. (Opsional) Masukkan deskripsi untuk deskripsi Deployment.
4. Pilih Deploy untuk mendorong versi API terbaru ke rilis canary.
5. Jika diinginkan, konfigurasi ulang pengaturan panggung, log, atau pengaturan kenari, seperti yang dijelaskan dalam. [Untuk membuat penerapan rilis kenari awal](#)

Akibatnya, rilis kenari menunjuk ke versi terbaru sementara rilis produksi masih menunjuk ke versi awal API. [CanarySettings](#) sekarang memiliki nilai `deploymentID` baru, sedangkan stage masih memiliki nilai `deploymentID` awal. Di belakang layar, konsol memanggil `stage:update`.

Buat penerapan kenari menggunakan AWS CLI

Untuk membuat penyebaran kenari untuk tahap baru

1. Gunakan perintah [create-deployment](#) berikut untuk membuat deployment dengan dua variabel tahap, tetapi tanpa canary:

```
aws apigateway create-deployment \  
  --variables sv0=val0,sv1=val1 \  
  --rest-api-id abcd1234 \  
  --stage-name 'prod'
```

Outputnya akan terlihat seperti berikut:

```
{  
  "id": "du4ot1",  
  "createdDate": 1511379050  
}
```

2. Gunakan perintah [create-deployment](#) berikut untuk membuat penyebaran kenari di panggung: `prod`

```
aws apigateway create-deployment \  
  --rest-api-id abcd1234 \  
  --canary-settings  
  percentTraffic=10.5,stageVariableOverrides={sv1='val2',sv2='val3'},useStageCache=false  
  \  
  --stage-name 'prod'
```

Outputnya akan terlihat seperti berikut:

```
{  
  "id": "a6rox0",  
  "createdDate": 1511379433  
}
```

Penerapan yang dihasilkan id mengidentifikasi versi uji API untuk rilis canary. Akibatnya, tahap terkait diaktifkan kenari.

3. (Opsional) Gunakan perintah [get-stage](#) berikut untuk melihat representasi tahap:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Berikut ini menunjukkan representasi dari Stage sebagai output dari perintah:

```
{
  "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": {}
}
```

Dalam contoh ini, versi dasar API akan menggunakan variabel tahap{"sv0":val0", "sv1":val1"}, sedangkan versi pengujian menggunakan variabel tahap{"sv1":val2", "sv2":val3"}. Baik rilis produksi dan rilis kenari menggunakan variabel tahap yang sama sv1, tetapi dengan nilai yang berbeda, val1 dan val2, masing-masing. Variabel tahap sv0 digunakan hanya dalam rilis produksi dan variabel tahap sv2 digunakan dalam rilis kenari saja.

Anda juga dapat membuat penerapan rilis kenari dari penerapan reguler yang ada dengan memperbarui tahapan untuk mengaktifkan kenari.

Untuk membuat penerapan rilis kenari dari penerapan yang ada

1. Gunakan perintah [create-deployment](#) untuk membuat deployment tanpa canary:

```
aws apigateway create-deployment \  
  --variables sv0=val0,sv1=val1 \  
  --rest-api-id abcd1234 \  
  --stage-name 'beta'
```

2. Gunakan perintah [update-stage](#) untuk memperbarui stage untuk mengaktifkan kenari:

```
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"  
  } ]'
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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,  
    "stageVariableOverrides": {  
        "sv2": "val3",  
        "sv1": "val2"  
    }  
  },  
  "methodSettings": {}  
}
```

Karena Anda mengaktifkan canary pada versi API yang ada, baik production release (Stage) dan canary release (canarySettings) mengarah ke penerapan yang sama. Setelah Anda mengubah API dan menerapkannya ke tahap ini lagi, versi baru akan berada di rilis canary, sementara versi dasar tetap dalam rilis produksi. Ini dimanifestasikan dalam evolusi tahap ketika rilis deploymentId dalam kenari diperbarui ke penerapan baru id dan rilis produksi tetap tidak berubah. deploymentId

Perbarui rilis kenari

Setelah rilis kenari diterapkan, Anda mungkin ingin menyesuaikan persentase lalu lintas kenari atau mengaktifkan atau menonaktifkan penggunaan cache panggung untuk mengoptimalkan kinerja pengujian. Anda juga dapat memodifikasi variabel tahap yang digunakan dalam rilis kenari saat konteks eksekusi diperbarui. [Untuk melakukan pembaruan seperti itu, panggil operasi stage:update dengan nilai baru di CanarySettings.](#)

Anda dapat memperbarui rilis canary menggunakan konsol API Gateway, perintah AWS CLI [tahap pembaruan](#), atau [SDK](#). AWS

Topik

- [Memperbarui rilis kenari menggunakan konsol API Gateway](#)
- [Perbarui rilis kenari menggunakan AWS CLI](#)

Memperbarui rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk memperbarui setelan canary yang ada di panggung, lakukan hal berikut:

Untuk memperbarui pengaturan kenari yang ada

1. Masuk ke konsol API Gateway dan pilih REST API yang ada.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary, lalu pilih Edit. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Canary.
4. Perbarui distribusi Permintaan dengan menambah atau mengurangi jumlah persentase antara 0,0 dan 100,0, inklusif.
5. Pilih atau hapus Stage cache kotak centang.
6. Menambahkan, menghapus, atau memodifikasi variabel tahap Canary.
7. Pilih Simpan.

Perbarui rilis kenari menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk memperbarui kenari, gunakan [update-stage](#) perintah dan modifikasi operasi tambalan untuk setiap parameter kenari.

Perintah [update-stage](#) berikut diperbarui jika kenari menggunakan cache panggung:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Perintah [tahap pembaruan berikut memperbarui](#) persentase lalu lintas kenari:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

[Pembaruan-tahap berikut memperbarui variabel tahap tahap](#). Contoh ini menunjukkan cara membuat variabel stage baru bernamanewVar, mengganti variabel var2 stage, dan menghapus variabel var1 stage:

```
aws apigateway update-stage \  
  --rest-api-id {rest-api-id} \  
  --stage-name '{stage-name}' \  
  --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

```
--rest-api-id {rest-api-id} \
--stage-name '{stage-name}' \
--patch-operations ' [{
  "op": "replace",
  "path": "/canarySettings/stageVariable0verrides/newVar",
  "value": "newVal"
}, {
  "op": "replace",
  "path": "/canarySettings/stageVariable0verrides/var2",
  "value": "val4"
}, {
  "op": "remove",
  "path": "/canarySettings/stageVariable0verrides/var1"
}]'
```

Anda dapat memperbarui semua hal di atas dengan menggabungkan operasi menjadi satu patch-operations nilai:

```
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/stageVariable0verrides/var1"
}, {
  "op": "replace",
  "path": "/canarySettings/stageVariable0verrides/newVar",
  "value": "newVal"
}, {
  "op": "replace",
  "path": "/canarySettings/stageVariable0verrides/val2",
  "value": "val4"
}]'
```

Promosikan pelepasan kenari

Saat Anda mempromosikan rilis kenari, rilis kenari menggantikan pengaturan tahap saat ini. Mempromosikan pelepasan kenari tidak menonaktifkan kenari di atas panggung. Untuk menonaktifkan kenari, Anda harus menghapus pengaturan kenari di atas panggung. Untuk mempromosikan kenari, lakukan hal berikut.

- Setel ulang [ID penyebaran](#) panggung dengan pengaturan [ID penerapan](#) kenari. Ini memperbarui snapshot API panggung dengan snapshot kenari, menjadikan versi uji sebagai rilis produksi juga.
- Perbarui variabel tahap dengan variabel tahap kenari, jika ada. Ini memperbarui konteks eksekusi API panggung dengan konteks kenari. Tanpa pembaruan ini, versi API baru dapat menghasilkan hasil yang tidak terduga jika versi pengujian menggunakan variabel tahap yang berbeda atau nilai berbeda dari variabel tahap yang ada.
- Atur persentase lalu lintas kenari menjadi 0,0%.

Topik

- [Promosikan rilis kenari menggunakan konsol API Gateway](#)
- [Promosikan pelepasan kenari menggunakan AWS CLI](#)

Promosikan rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mempromosikan penerapan rilis canary, lakukan hal berikut:

Untuk mempromosikan penyebaran rilis kenari

1. Masuk ke konsol API Gateway dan pilih API yang ada di panel navigasi utama.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary.
4. Pilih Promosikan kenari.
5. Konfirmasikan perubahan yang akan dilakukan dan pilih Promosikan kenari.

Setelah promosi, rilis produksi mereferensikan versi API yang sama (deploymentID) dengan rilis canary. Anda dapat memverifikasi ini menggunakan AWS CLI. Sebagai contoh, lihat [the section called “Promosikan pelepasan kenari menggunakan AWS CLI”](#).

Promosikan pelepasan kenari menggunakan AWS CLI

Untuk mempromosikan rilis kenari ke rilis produksi menggunakan AWS CLI perintah, panggil `update-stage` perintah untuk menyalin kenari yang terkait dengan tahap terkait, `deploymentId` untuk mengatur ulang persentase lalu lintas kenari ke nol (`0.0`), `deploymentId`, dan, untuk menyalin variabel tahap terikat kenari ke yang terikat tahap yang sesuai.

Misalkan kita memiliki penyebaran rilis kenari, dijelaskan oleh tahap yang mirip dengan yang berikut ini:

```
{
  "_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": "nfcn0x",
  "lastUpdatedDate": "2017-11-22T00:54:28Z",
  "methodSettings": {
    ...
  },
  "stageName": "prod",
  "variables": {
    "sv1": "val1"
  }
}
```

Gunakan perintah [update-stage](#) berikut untuk mempromosikan kenari:

```
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/stageVariable0verrides",
  "path": "/variables"
}, {
  "op": "copy",
  "from": "/canarySettings/deploymentId",
  "path": "/deploymentId"
}]'
```

Outputnya akan terlihat seperti berikut:

```
{
  "_links": {
    ...
  },
  "accessLogSettings": {
    ...
  },
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "canarySettings": {
    "deploymentId": "eh1sby",
    "useStageCache": false,
    "stageVariable0verrides": {
      "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"  
}  
}
```

Mempromosikan rilis kenari ke panggung tidak menonaktifkan kenari dan penerapan tetap menjadi penerapan rilis kenari. Untuk menjadikannya penyebaran rilis produksi reguler, Anda harus menonaktifkan pengaturan kenari. Untuk informasi selengkapnya tentang cara menonaktifkan penerapan rilis kenari, lihat. [the section called “Matikan pelepasan kenari”](#)

Matikan pelepasan kenari

Untuk mematikan penerapan rilis kenari berarti menyetel ke null [canarySettings](#) untuk menghapusnya dari panggung.

Anda dapat menonaktifkan penerapan rilis canary menggunakan konsol API Gateway, the AWS CLI, atau SDK. AWS

Topik

- [Matikan rilis kenari menggunakan konsol API Gateway](#)
- [Matikan pelepas kenari menggunakan AWS CLI](#)

Matikan rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mematikan penerapan rilis canary, gunakan langkah-langkah berikut:

Untuk mematikan penerapan rilis kenari

1. Masuk ke konsol API Gateway dan pilih API yang ada di panel navigasi utama.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary.
4. Pilih Hapus.
5. Konfirmasikan bahwa Anda ingin menghapus kenari dengan memilih Hapus.

Akibatnya, [canarySettings](#) properti menjadi null dan dihapus dari [tahap](#) penyebaran. Anda dapat memverifikasi ini menggunakan AWS CLI. Sebagai contoh, lihat [the section called “Matikan pelepas kenari menggunakan AWS CLI”](#).

Matikan pelepas kenari menggunakan AWS CLI

Perintah [update-stage](#) berikut mematikan penerapan rilis kenari:

```
aws apigateway update-stage \  
  --rest-api-id abcd1234 \  
  --stage-name canary \  
  --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

Outputnya terlihat seperti berikut:

```
{  
  "stageName": "prod",  
  "accessLogSettings": {  
    ...  
  },  
  "cacheClusterEnabled": false,  
  "cacheClusterStatus": "NOT_AVAILABLE",  
  "deploymentId": "nfcn0x",  
  "lastUpdatedDate": 1511309280,  
  "createdDate": 1511152939,  
  "methodSettings": {  
    ...  
  }  
}
```

[Seperti yang ditunjukkan dalam output, properti CanarySettings tidak lagi hadir dalam tahap penerapan canary-disabled.](#)

Pembaruan untuk REST APIs yang memerlukan pemindahan

Mempertahankan API berarti melihat, memperbarui, dan menghapus pengaturan API yang ada. Anda dapat mempertahankan API menggunakan konsol API Gateway, AWS CLI, SDK CloudFormation, atau API Gateway REST API. Memperbarui API melibatkan modifikasi properti sumber daya tertentu atau pengaturan konfigurasi API. Pembaruan sumber daya memerlukan penerapan ulang API, di mana pembaruan konfigurasi tidak.

Tabel berikut menjelaskan resource API yang memerlukan redeployment API Anda saat Anda memperbaruinya.

Sumber daya	Catatan
ApiKey	Untuk properti yang berlaku dan operasi yang didukung, lihat apikey:update . Pembaruan memerlukan penerapan ulang API.
Pengotorisasi	Untuk properti yang berlaku dan operasi yang didukung, lihat authorize:update . Pembaruan memerlukan penerapan ulang API.
disableExecuteApiTitikakhir	Pembaruan memerlukan modifikasi tahap apa pun pada API seperti memindahkan API ke panggung.
DocumentationPart	Untuk properti yang berlaku dan operasi yang didukung, lihat documentationpart:update . Pembaruan memerlukan penerapan ulang API.
DocumentationVersion	Untuk properti yang berlaku dan operasi yang didukung, lihat documentationversion:update . Pembaruan memerlukan penerapan ulang API.
GatewayResponse	Untuk properti yang berlaku dan operasi yang didukung, lihat gatewayresponse:update . Pembaruan memerlukan penerapan ulang API.
Integrasi	Untuk properti yang berlaku dan operasi yang didukung, lihat integration:update . Pembaruan memerlukan penerapan ulang API.
IntegrationResponse	Untuk properti yang berlaku dan operasi yang didukung, lihat integrationresponse:update . Pembaruan memerlukan penerapan ulang API.
Metode	Untuk properti yang berlaku dan operasi yang didukung, lihat metode:update . Pembaruan memerlukan penerapan ulang API.
MethodResponse	Untuk properti yang berlaku dan operasi yang didukung, lihat methodresponse:update . Pembaruan memerlukan penerapan ulang API.
Model	Untuk properti yang berlaku dan operasi yang didukung, lihat model:update . Pembaruan memerlukan penerapan ulang API.

Sumber daya	Catatan
RequestValidator	Untuk properti yang berlaku dan operasi yang didukung, lihat requestvalidator:update . Pembaruan memerlukan penerapan ulang API.
Sumber Daya	Untuk properti yang berlaku dan operasi yang didukung, lihat resource:update . Pembaruan memerlukan penerapan ulang API.
RestApi	Untuk properti yang berlaku dan operasi yang didukung, lihat restapi:update . Pembaruan memerlukan penerapan ulang API. Ini termasuk memodifikasi kebijakan sumber daya.
VpcLink	Untuk properti yang berlaku dan operasi yang didukung, lihat vpclink:update . Pembaruan memerlukan penerapan ulang API.

Tabel berikut menjelaskan konfigurasi API yang tidak memerlukan redeployment API Anda saat Anda memperbaruinya.

Konfigurasi	Catatan
Akun	Untuk properti yang berlaku dan operasi yang didukung, lihat account:update . Pembaruan tidak memerlukan penerapan ulang API.
Deployment	Untuk properti yang berlaku dan operasi yang didukung, lihat deployment:update .
DomainName	Untuk properti yang berlaku dan operasi yang didukung, lihat domainname:update . Pembaruan tidak memerlukan penerapan ulang API.
BasePathMapping	Untuk properti yang berlaku dan operasi yang didukung, lihat basepathmapping:update . Pembaruan tidak memerlukan penerapan ulang API.
Jenis alamat IP	Pembaruan tidak memerlukan penerapan ulang API.
Panggung	Untuk properti yang berlaku dan operasi yang didukung, lihat stage:update . Pembaruan tidak memerlukan penerapan ulang API.

Konfigurasi	Catatan
Penggunaan	Untuk properti yang berlaku dan operasi yang didukung, lihat usage:update . Pembaruan tidak memerlukan penerapan ulang API.
UsagePlan	Untuk properti yang berlaku dan operasi yang didukung, lihat usageplan:update . Pembaruan tidak memerlukan penerapan ulang API.

Nama domain khusus untuk REST publik APIs di API Gateway

Nama domain khusus lebih sederhana dan lebih intuitif URLs yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API, Anda (dan pelanggan) dapat menjalankan API menggunakan URL dasar default dari format berikut:

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

dimana *api-id* dihasilkan oleh API Gateway, *region* adalah AWS Wilayah, dan *stage* ditentukan oleh Anda saat menerapkan API.

Bagian nama host dari URL, *api-id*.execute-api.*region*.amazonaws.com mengacu pada titik akhir API. Nama endpoint API default dibuat secara acak, sulit diingat, dan tidak ramah pengguna.

Dengan nama domain khusus, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya, *myservice*) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

Note

Untuk informasi tentang nama domain khusus untuk pribadi APIs, lihat [the section called “Nama domain khusus untuk pribadi APIs”](#).

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan nama domain kustom Anda:

- Anda dapat menonaktifkan titik akhir default untuk API Anda. Klien masih dapat terhubung ke titik akhir default Anda, tetapi mereka akan menerima kode 403 Forbidden status.
- Nama domain kustom Regional dapat dikaitkan dengan REST APIs dan HTTP APIs. Anda dapat menggunakan [API Gateway Versi 2 APIs](#) untuk membuat dan mengelola nama domain kustom Regional untuk REST APIs.
- Nama domain khusus harus unik di dalam Wilayah di semua AWS akun.
- Anda dapat memigrasikan nama domain kustom antara titik akhir yang dioptimalkan tepi dan Regional, tetapi Anda tidak dapat memigrasikan domain kustom publik ke nama domain kustom pribadi.
- Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS Anda untuk memetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain khusus tidak dapat mencapai API Gateway.
- Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default dengan menggunakan sertifikat wildcard. Untuk informasi selengkapnya, lihat [the section called “Nama domain kustom wildcard”](#).
- Anda dapat memilih kebijakan keamanan untuk nama domain kustom Anda. Untuk informasi selengkapnya, lihat [the section called “Pilih kebijakan keamanan”](#).
- Untuk mengonfigurasi pemetaan API dengan beberapa level, Anda harus menggunakan nama domain kustom Regional dan menggunakan kebijakan keamanan TLS 1.2.

Prasyarat untuk nama domain khusus

Berikut ini adalah prasyarat untuk membuat nama domain kustom publik atau pribadi. Untuk informasi tentang nama domain khusus untuk pribadi APIs, lihat [the section called “Nama domain khusus untuk pribadi APIs”](#).

Daftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk mengatur nama domain khusus untuk Anda APIs. Anda dapat mendaftarkan nama domain internet Anda menggunakan [Amazon Route 53](#) atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom Anda dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zone apex”) dari domain internet terdaftar.

Nama domain Anda harus mengikuti spesifikasi [RFC 1035](#) dan dapat memiliki maksimum 63 oktet per label dan total 255 oktet.

Tentukan sertifikat untuk nama domain kustom Anda

Sebelum menyiapkan nama domain khusus untuk API, Anda harus memiliki SSL/TLS sertifikat yang siap di ACM. Jika ACM tidak tersedia di AWS Wilayah tempat Anda membuat nama domain kustom, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor SSL/TLS sertifikat, Anda harus menyediakan badan SSL/TLS sertifikat berformat PEM, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom.

Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir mengekspos detail sertifikat sensitif, seperti kunci pribadi. Untuk menggunakan sertifikat AWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

Jika aplikasi Anda menggunakan pinning sertifikat, kadang-kadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelah AWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat [Masalah penyematan sertifikat](#) di Panduan AWS Certificate Manager Pengguna.

Nama domain kustom wildcard

Dengan nama domain khusus wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota [default](#). Misalnya, Anda bisa memberi setiap pelanggan Anda nama domain mereka sendiri `customername.example.com`.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (*) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard `*.example.com` menghasilkan subdomain seperti `example.com`, `b.example.com` dan `c.example.com`. Saat Anda membuat nama domain khusus wildcard, semua subdomainnya dirutekan oleh mode perutean nama domain wildcard. Untuk merutekan subdomain ke yang berbeda APIs, Anda dapat melakukan salah satu hal berikut:

- Gunakan aturan perutean untuk merutekan permintaan masuk `*.example.com` ke REST target yang berbeda APIs menggunakan header. Host Untuk informasi selengkapnya, lihat [the section called “Contoh 4: Aturan perutean untuk nama domain wildcard”](#).
- Buat nama domain untuk setiap subdomain yang ingin Anda rute ke titik akhir yang berbeda. Dalam satu AWS akun, Anda dapat memiliki keduanya `*.example.com` dan `example.com`.

Anda dapat menggunakan variabel `$context.domainName` dan `$context.domainPrefix` konteks untuk menentukan nama domain yang digunakan klien untuk memanggil API Anda. Untuk mempelajari lebih lanjut tentang variabel konteks, lihat [Variabel untuk transformasi data untuk API Gateway](#).

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

Note

Anda tidak dapat membuat nama domain khusus wildcard jika AWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuat `.example.com`, maka akun B tidak dapat membuat nama `*.example.com` domain khusus wildcard.

Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi [Pusat AWS Dukungan](#) untuk meminta pengecualian.

Langkah selanjutnya untuk nama domain kustom

Berikut ini adalah langkah selanjutnya untuk nama domain kustom.

Langkah selanjutnya

- Untuk mempelajari cara mengatur SSL/TLS sertifikat Anda, lihat [the section called “Siapkan sertifikat AWS Certificate Manager”](#).
- Untuk mempelajari cara membuat nama domain kustom Regional, lihat [the section called “Siapkan nama domain kustom Regional”](#).
- Untuk mempelajari cara membuat nama domain kustom yang dioptimalkan tepi, lihat [the section called “Siapkan nama domain kustom yang dioptimalkan tepi di API Gateway”](#).
- Untuk mempelajari cara bermigrasi antara nama domain kustom Regional dan yang dioptimalkan di tepi, lihat [the section called “Migrasi nama domain kustom”](#).
- Untuk mempelajari cara menghubungkan tahapan API ke nama domain kustom, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).
- Untuk mempelajari cara memilih kebijakan keamanan untuk nama domain kustom Anda, lihat [the section called “Pilih kebijakan keamanan”](#).

- Untuk mempelajari cara menonaktifkan titik akhir default untuk nama domain kustom Anda, lihat [the section called “Nonaktifkan titik akhir default”](#).
- Untuk mempelajari cara menggunakan pemeriksaan kesehatan Route 53 untuk mengontrol failover DNS dari API Gateway API, lihat [the section called “failover DNS”](#)

Jika ini adalah pertama kalinya Anda membuat nama domain kustom, kami sarankan Anda mulai dengan [the section called “Siapkan sertifikat AWS Certificate Manager”](#), untuk menentukan sertifikat Anda, dan kemudian [the section called “Siapkan nama domain kustom Regional”](#) membuat nama domain kustom Regional.

Siapkan sertifikat AWS Certificate Manager

Sebelum menyiapkan nama domain khusus untuk API, Anda harus memiliki sertifikat SSL/TLS yang siap. AWS Certificate Manager Untuk informasi selengkapnya, silakan lihat [Panduan Pengguna AWS Certificate Manager](#).

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk SSL/TLS sertifikat Anda.

- Jika Anda membuat nama domain kustom yang dioptimalkan tepi, API Gateway memanfaatkan CloudFront untuk mendukung sertifikat untuk nama domain kustom. Dengan demikian, persyaratan dan batasan SSL/TLS sertifikat nama domain kustom ditentukan oleh [CloudFront](#). Misalnya, ukuran maksimum kunci publik adalah 2048 dan ukuran kunci pribadi bisa 1024, 2048, dan 4096. Ukuran kunci publik ditentukan oleh otoritas sertifikat yang Anda gunakan. Minta otoritas sertifikat Anda untuk mengembalikan kunci dengan ukuran yang berbeda dari panjang default. Untuk informasi selengkapnya, lihat [Akses aman ke objek Anda](#) dan [Buat cookie yang ditandatangani URLs dan ditandatangani](#).
- Jika Anda membuat nama domain kustom Regional, ukuran maksimum kunci publik adalah 2048.
- Untuk menggunakan sertifikat ACM dengan nama domain kustom Regional, Anda harus meminta atau mengimpor sertifikat di Wilayah yang sama dengan API Anda. Sertifikat harus mencakup nama domain khusus.
- Untuk menggunakan sertifikat ACM dengan nama domain kustom yang dioptimalkan tepi, Anda harus meminta atau mengimpor sertifikat di Wilayah AS Timur (Virginia N.). us-east-1
- Anda harus memiliki nama domain terdaftar, seperti *example.com*. Anda dapat menggunakan [Amazon Route 53](#) atau registrar domain terakreditasi pihak ketiga. Untuk daftar pendaftar tersebut, lihat [Direktori Panitera Terakreditasi di situs web ICANN](#).

Untuk membuat atau mengimpor SSL/TLS sertifikat ke ACM

Prosedur berikut menunjukkan cara membuat atau mengimpor SSL/TLS sertifikat untuk nama domain.

To request a certificate provided by ACM for a domain name

1. Masuk ke [konsol AWS Certificate Manager](#) tersebut.
2. Pilih Minta sertifikat.
3. Untuk jenis Sertifikat, pilih Minta sertifikat publik.
4. Pilih Berikutnya.
5. Untuk nama domain yang sepenuhnya memenuhi syarat, masukkan nama domain khusus untuk API Anda, misalnya, `api.example.com`.
6. Secara opsional, pilih Tambahkan nama lain ke sertifikat ini.
7. Untuk metode Validasi, pilih metode untuk memvalidasi kepemilikan domain.
8. Untuk Algoritma kunci, pilih algoritma enkripsi.
9. Pilih Minta.
10. Untuk permintaan yang valid, pemilik domain internet yang terdaftar harus menyetujui permintaan tersebut sebelum ACM mengeluarkan sertifikat. Jika Anda menggunakan Route 53 untuk mengelola catatan DNS publik, Anda dapat memperbarui catatan Anda melalui konsol ACM secara langsung.

To import into ACM a certificate for a domain name

1. Dapatkan sertifikat yang dikodekan PEM untuk nama domain kustom Anda dari otoritas SSL/TLS sertifikat. Untuk daftar parsialnya CAs, lihat [Daftar CA Termasuk Mozilla](#).
 - a. Buat kunci pribadi untuk sertifikat dan simpan output ke file, menggunakan toolkit [OpenSSL di situs web](#) OpenSSL:

```
openssl genrsa -out private-key-file 2048
```
 - b. Buat permintaan penandatanganan sertifikat (CSR) dengan kunci pribadi yang dibuat sebelumnya, menggunakan OpenSSL:

```
openssl req -new -sha256 -key private-key-file -out CSR-file
```

- c. Kirimkan CSR ke otoritas sertifikat dan simpan sertifikat yang dihasilkan.
- d. Unduh rantai sertifikat dari otoritas sertifikat.

 Note

Jika Anda mendapatkan kunci pribadi dengan cara lain dan kunci dienkripsi, Anda dapat menggunakan perintah berikut untuk mendekripsi kunci sebelum mengirimkannya ke API Gateway untuk menyiapkan nama domain kustom.

```
openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
nocrypt -out MyDecryptedKey.pem
```

2. Unggah sertifikat ke AWS Certificate Manager:

- a. Masuk ke [konsol AWS Certificate Manager](#) tersebut.
- b. Pilih Impor sertifikat.
- c. Untuk badan Sertifikat, masukkan badan sertifikat server berformat PEM dari otoritas sertifikat Anda. Berikut ini menunjukkan contoh singkat dari sertifikat tersebut.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8P1++g0fQtj0IBoqDANBgqhkiG9w0BAQUFADBB
...
az8Cg1a1cxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

- d. Untuk kunci pribadi Sertifikat, masukkan kunci pribadi sertifikat berformat PEM Anda. Berikut ini menunjukkan contoh singkat dari kunci tersebut.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nET0
...
1qGvJ3u04vdnzaYN5WoyN5LFckr1A71+CszD1CGSqBVdWEXAMPLE
-----END RSA PRIVATE KEY-----
```

- e. Untuk rantai Sertifikat, masukkan sertifikat perantara berformat PEM dan, secara opsional, sertifikat root, satu demi satu tanpa baris kosong. Jika Anda menyertakan sertifikat root, rantai sertifikat Anda harus dimulai dengan sertifikat perantara dan diakhiri dengan sertifikat root. Gunakan sertifikat perantara yang disediakan oleh

otoritas sertifikat Anda. Jangan sertakan perantara yang tidak berada dalam jalur rantai kepercayaan. Berikut ini menunjukkan contoh singkat.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifB1IK3se2e4/hEfcEejX/arxbx1BJCHBv1EPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Berikut adalah contoh lain.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Pilih Berikutnya, lalu pilih Berikutnya.

Setelah sertifikat berhasil dibuat atau diimpor, catat sertifikat ARN. Anda membutuhkannya saat menyiapkan nama domain khusus.

Siapkan nama domain kustom Regional di API Gateway

Gunakan nama domain kustom Regional untuk membuat URL dasar API yang mudah digunakan. Dengan nama domain kustom Regional, Anda dapat memetakan tahap HTTP dan REST API ke nama domain kustom yang sama dan menggunakan otentikasi TLS bersama.

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk nama domain kustom Regional Anda:

- Anda harus memberikan sertifikat ACM khusus Wilayah. Sertifikat ini harus berada di Wilayah yang sama dengan API Anda. Untuk informasi selengkapnya tentang membuat atau mengunggah sertifikat nama domain kustom, lihat [the section called “Siapkan sertifikat AWS Certificate Manager”](#).

- Saat Anda membuat nama domain kustom Regional (atau memigrasikan satu) dengan sertifikat ACM, API Gateway akan membuat peran terkait layanan di akun Anda. Peran terkait layanan diperlukan untuk melampirkan sertifikat ACM Anda ke titik akhir Regional Anda. Peran tersebut diberi nama `AWSServiceRoleForAPIGateway` dan akan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang penggunaan peran terkait layanan, lihat [Menggunakan Peran Tertaut Layanan](#).
- Setelah Anda membuat nama domain kustom Regional Anda, Anda harus membuat catatan DNS untuk mengarahkan nama domain kustom ke domain Regional. Hal ini memungkinkan lalu lintas yang terikat pada nama domain kustom untuk dirutekan ke nama host Regional API.

Catatan DNS dapat berupa catatan CNAME atau A Alias. Jika Anda menggunakan Route 53 sebagai penyedia DNS Anda, buat catatan Alias A. Jika Anda menggunakan penyedia DNS pihak ketiga, gunakan catatan CNAME. Jika Anda menggunakan catatan CNAME dan membuat titik akhir VPC antarmuka API Gateway dengan DNS pribadi diaktifkan untuk API pribadi, Anda tidak dapat menyelesaikan nama domain kustom dalam VPC yang menghosting API pribadi Anda.

Buat nama domain kustom Regional

Prosedur berikut menunjukkan cara membuat nama domain kustom Regional. Setelah menyelesaikan prosedur ini, Anda membuat aturan perutean untuk merutekan tahapan API Anda ke nama domain kustom Anda.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih Buat.
4. Untuk nama Domain, masukkan nama domain.
5. Untuk mode Routing, pilih Routing rules only.

Dalam mode routing ini, Anda hanya dapat mengirim lalu lintas dari nama domain kustom Anda ke Anda APIs dengan menggunakan aturan routing. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

6. Untuk versi TLS Minimum, pilih versi.
7. Di bawah Konfigurasi titik akhir, untuk jenis titik akhir API, pilih Regional.

8. Pilih sertifikat ACM. Sertifikat harus berada di Wilayah yang sama dengan API.
9. Pilih Buat.

AWS CLI

[create-domain-name](#) Perintah berikut membuat nama domain kustom:

```
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
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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"  
}
```

Nilai `DomainNameConfigurations` properti mengembalikan nama host API Regional. Anda harus membuat catatan DNS untuk mengarahkan nama domain kustom Anda ke nama domain Regional ini. Ini memungkinkan lalu lintas yang terikat pada nama domain kustom untuk dirutekan ke nama host API Regional ini.

Buat aturan perutean untuk nama domain kustom Regional Anda

Setelah Anda membuat nama domain kustom Anda, Anda mengonfigurasi bagaimana lalu lintas dirutekan dari nama domain khusus Anda ke nama domain Anda APIs. Karena Anda mengatur mode perutean `ROUTING_RULE_ONLY`, Anda menggunakan aturan perutean untuk merutekan permintaan masuk ke nama domain kustom Anda ke nama domain Anda. APIs

Dalam contoh ini, Anda membuat aturan catch-all yang merutekan semua permintaan masuk ke nama domain kustom Anda ke satu tahap API Anda. Anda juga dapat mengonfigurasi aturan perutean berdasarkan kondisi header dan jalur yang berbeda. Untuk informasi selengkapnya, lihat [the section called “Aturan perutean”](#).

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama domain khusus.
3. Pada tab Routing details, pilih Add routing rule.
4. Pilih Tambahkan kondisi baru untuk menambahkan kondisi baru.
5. Pertahankan aturan ini tanpa syarat apa pun. Ini merutekan semua permintaan ke nama domain kustom Anda ke API target dan tahap target Anda.
6. Untuk Action, gunakan dropdown untuk memilih API target dan tahap target Anda.
7. Pilih Berikutnya.
8. Di bidang prioritas, masukkan **100**.

API Gateway mengevaluasi aturan dalam urutan prioritas, dari nilai terendah hingga nilai tertinggi. Karena ini adalah aturan catch-all, Anda menggunakan prioritas tinggi sehingga API Gateway dapat mencocokkan aturan tambahan apa pun yang Anda buat terlebih dahulu.

9. Pilih Buat aturan perutean.

AWS CLI

`create-routing-rule` Perintah berikut membuat aturan perutean catch-all:

```
aws apigatewayv2 create-routing-rule \  
  --domain-name 'regional.example.com' \  
  --priority 100 \  
  --conditions \  
  --
```

```
--actions ' [{
  "InvokeApi": {
    "ApiId": "a1b2c3",
    "Stage": "prod"
  }
}]'
```

Anda dapat mengubah mode perutean dan membuat aturan baru kapan saja. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

Buat catatan DNS untuk nama domain kustom Regional Anda

Setelah Anda membuat nama domain kustom dan membuat pemetaan jalur dasar, Anda membuat catatan DNS untuk mengarahkan nama domain kustom Anda nama domain Regional yang baru dibuat.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, ikuti dokumentasi Route 53 tentang [mengonfigurasi Route 53 untuk merutekan lalu lintas ke API Gateway](#).

AWS CLI

Untuk mengonfigurasi catatan DNS Anda untuk memetakan nama domain kustom Regional ke nama hostnya dari ID zona host yang diberikan, pertama-tama buat file JSON yang berisi konfigurasi untuk menyiapkan catatan DNS untuk nama domain Regional.

Berikut ini `setup-dns-record.json` menunjukkan cara membuat A catatan DNS untuk memetakan nama domain kustom Regional (`regional.example.com`) ke nama host Regional (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) yang disediakan sebagai bagian dari pembuatan nama domain kustom. `HostedZoneId` properti `DNSName` dan `AliasTarget` dapat mengambil `regionalHostedZoneId` nilai `regionalDomainName` dan, masing-masing, dari nama domain kustom. Anda juga bisa mendapatkan Regional Route 53 Hosted Zone IDs di [Amazon API Gateway Endpoints dan Quotas](#).

```
{
  "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
      }
    }
  ]
}

```

Berikut ini [change-resource-record-sets](#) membuat catatan DNS untuk nama domain kustom Regional Anda:

```

aws route53 change-resource-record-sets \
  --hosted-zone-id Z20JLYMU09EFXC \
  --change-batch file://path/to/your/setup-dns-record.json

```

Ganti `hosted-zone-id` dengan ID Zona Dihosting Route 53 dari catatan DNS yang ditetapkan di akun Anda. Nilai `change-batch` parameter menunjuk ke file JSON (*setup-dns-record.json*) dalam folder (*path/to/your*).

Siapkan nama domain kustom yang dioptimalkan tepi di API Gateway

Saat Anda membuat nama domain khusus untuk API yang dioptimalkan tepi, API Gateway menyiapkan CloudFront distribusi dan catatan DNS untuk memetakan nama domain API ke nama domain CloudFront distribusi. Permintaan untuk API kemudian dirutekan ke API Gateway melalui distribusi yang dipetakan CloudFront. Pemetaan ini untuk permintaan API yang terikat untuk nama domain kustom yang akan dirutekan ke API Gateway melalui distribusi yang CloudFront dipetakan.

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk nama domain kustom yang dioptimalkan tepi Anda:

- Untuk menyiapkan nama domain kustom yang dioptimalkan tepi atau memperbarui sertifikatnya, Anda harus memiliki izin untuk memperbarui CloudFront distribusi.

Izin berikut diperlukan untuk memperbarui CloudFront distribusi:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontUpdateDistribution",
      "Effect": "Allow",
      "Action": [
        "cloudfront:updateDistribution"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- Anda harus meminta atau mengimpor sertifikat untuk nama domain kustom yang dioptimalkan tepi Anda di Wilayah AS Timur (Virginia N.). us-east-1
- CloudFront Distribusi yang dibuat oleh API Gateway dimiliki oleh akun khusus Wilayah yang berafiliasi dengan API Gateway. Saat melacak operasi untuk membuat dan memperbarui CloudFront distribusi seperti itu CloudTrail, Anda harus menggunakan ID akun API Gateway ini. Untuk informasi selengkapnya, lihat [Log pembuatan nama domain kustom di CloudTrail](#).
- API Gateway mendukung nama domain kustom yang dioptimalkan tepi dengan memanfaatkan Server Name Indication (SNI) pada distribusi. CloudFront Untuk informasi selengkapnya tentang penggunaan nama domain khusus pada CloudFront distribusi, termasuk format sertifikat yang diperlukan dan ukuran maksimum panjang kunci sertifikat, lihat [Menggunakan Nama Domain Alternatif dan HTTPS](#) di Panduan CloudFront Pengembang Amazon
- Nama domain kustom yang dioptimalkan tepi membutuhkan waktu sekitar 40 menit untuk siap.
- Setelah membuat nama domain kustom yang dioptimalkan tepi, Anda harus membuat catatan DNS untuk memetakan nama domain kustom ke nama distribusi. CloudFront

Buat nama domain kustom yang dioptimalkan tepi

Prosedur berikut menjelaskan cara membuat nama domain kustom yang dioptimalkan tepi untuk API.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih Tambahkan nama domain.
4. Untuk nama Domain, masukkan nama domain.
5. Untuk mode Routing, pilih API_MAPPING_ONLY.
6. Untuk jenis endpoint API, pilih Edge-optimized.
7. Pilih versi TLS minimum.
8. Pilih sertifikat ACM.
9. Pilih Tambahkan nama domain.

REST API

1. Panggil [domainname:create](#), tentukan nama domain kustom dan ARN dari sertifikat yang disimpan di. AWS Certificate Manager

Panggilan API yang berhasil mengembalikan 201 Created respons yang berisi ARN sertifikat serta nama CloudFront distribusi terkait dalam muatannya.

2. Perhatikan nama domain CloudFront distribusi yang ditunjukkan pada output. Anda memerlukannya di langkah berikutnya untuk mengatur target alias A-record domain kustom di DNS Anda.

Untuk contoh kode panggilan REST API ini, lihat [domainname:create](#).

Nama domain khusus yang dioptimalkan tepi membutuhkan waktu sekitar 40 menit untuk siap, tetapi konsol segera menampilkan nama domain CloudFront distribusi terkait, dalam bentuk *distribution-id*.cloudfront.net, bersama dengan sertifikat ARN. Sementara itu, Anda dapat membuat pemetaan jalur dasar atau aturan perutean dan kemudian mengonfigurasi alias catatan DNS untuk memetakan nama domain kustom ke nama domain distribusi terkait. CloudFront

Konfigurasi pemetaan jalur dasar API dengan nama domain khusus sebagai nama hostnya

Karena Anda mengatur mode perutean API_MAPPING_ONLY, Anda dapat menggunakan pemetaan jalur dasar untuk menggunakan satu nama domain kustom sebagai nama host dari beberapa APIs. Ini membuat API dapat diakses melalui kombinasi nama domain khusus dan jalur dasar terkait.

Misalnya, jika di API Gateway, Anda membuat API bernama PetStore dan API lain bernama Dogs lalu menyiapkan nama domain kustom `api.example.com`, Anda dapat mengatur URL PetStore API sebagai `https://api.example.com`.

Ini mengaitkan PetStore API dengan jalur dasar string kosong. Jika Anda menetapkan URL PetStore API sebagai `https://api.example.com/PetStore`, ini mengaitkan PetStore API dengan jalur dasar. PetStore Anda dapat menetapkan jalur dasar MyDogList untuk Dogs API. URL kemudian `https://api.example.com/MyDogList` adalah URL root dari Dogs API.

Untuk mengonfigurasi pemetaan API di beberapa level, Anda hanya dapat menggunakan nama domain kustom Regional. Nama domain kustom yang dioptimalkan tepi tidak didukung. Untuk informasi selengkapnya, lihat [the section called “Pemetaan API”](#).

Prosedur berikut menyiapkan pemetaan API untuk memetakan jalur dari nama domain kustom Anda ke tahapan API Anda.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama konsol API Gateway.
3. Pilih nama domain khusus.
4. Pilih Konfigurasi pemetaan API.
5. Pilih Tambahkan pemetaan baru.
6. Tentukan API, Stage, dan Path (opsional) untuk pemetaan.
7. Pilih Simpan.

REST API

Panggil [basepathmapping:create](#) pada nama domain kustom tertentu, tentukan `propertirestApiId`, dan deployment dalam `basePath` payload permintaan. stage

Panggilan API yang berhasil mengembalikan `201 Created` respons.

Untuk contoh kode panggilan REST API, lihat [basepathmapping:create](#).

Untuk fleksibilitas lebih lanjut tentang cara Anda merutekan lalu lintas ke Anda APIs, Anda dapat mengubah mode perutean ke `ROUTING_RULE_ONLY` atau `ROUTING_RULE_THEN_API_MAPPING`

dan membuat aturan perutean. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

Buat catatan DNS untuk nama domain kustom yang dioptimalkan tepi

Setelah Anda memulai pembuatan nama domain kustom yang dioptimalkan tepi, siapkan alias catatan DNS.

Kami menyarankan Anda menggunakan Route 53 untuk membuat alias A-record untuk nama domain kustom Anda dan menentukan nama domain CloudFront distribusi sebagai target alias. Ini berarti bahwa Route 53 dapat merutekan nama domain kustom Anda meskipun itu adalah puncak zona. Untuk informasi selengkapnya, lihat [Memilih Antara Kumpulan Rekaman Sumber Daya Alias dan Non-Alias di Panduan Pengembang](#) Amazon Route 53.

Untuk petunjuk tentang Amazon Route 53, lihat [Merutekan lalu lintas ke API Amazon API Gateway dengan menggunakan nama domain Anda](#) di Panduan Pengembang Amazon Route 53.

Log pembuatan nama domain kustom di CloudTrail

Saat CloudTrail diaktifkan untuk mencatat panggilan API Gateway yang dilakukan oleh akun Anda, API Gateway mencatat pembaruan CloudFront distribusi terkait saat nama domain khusus dibuat atau diperbarui untuk API. Log ini tersedia di us-east-1. Karena CloudFront distribusi ini dimiliki oleh API Gateway, masing-masing CloudFront distribusi yang dilaporkan diidentifikasi oleh salah satu akun API Gateway khusus Wilayah berikut, bukan ID IDs akun pemilik API.

Wilayah	ID Akun
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

Wilayah	ID Akun
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
me-central-1	614065512851

Putar sertifikat yang diimpor ke ACM

ACM secara otomatis menangani pembaruan sertifikat yang dikeluarkannya. Anda tidak perlu memutar sertifikat yang dikeluarkan ACM untuk nama domain kustom Anda. CloudFront menanganinya atas nama Anda.

Namun, jika Anda mengimpor sertifikat ke ACM dan menggunakannya untuk nama domain khusus, Anda harus memutar sertifikat sebelum kedaluwarsa. Ini melibatkan mengimpor sertifikat pihak ketiga baru untuk nama domain dan memutar sertifikat yang ada ke yang baru. Anda perlu mengulangi proses ketika sertifikat yang baru diimpor kedaluwarsa. Atau, Anda dapat meminta ACM untuk mengeluarkan sertifikat baru untuk nama domain dan memutar yang sudah ada ke sertifikat baru yang dikeluarkan ACM. Setelah itu, Anda dapat meninggalkan ACM dan CloudFront menangani rotasi sertifikat untuk Anda secara otomatis. Untuk membuat atau mengimpor sertifikat ACM baru, ikuti langkah-langkahnya. [the section called “Untuk membuat atau mengimpor SSL/TLS sertifikat ke ACM”](#)

Prosedur berikut menjelaskan cara memutar sertifikat untuk nama domain.

Note

Dibutuhkan sekitar 40 menit untuk memutar sertifikat yang diimpor ke ACM.

Konsol Manajemen AWS

1. Meminta atau mengimpor sertifikat di ACM.
2. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
3. Pilih Nama domain khusus dari panel navigasi utama konsol API Gateway.
4. Pilih nama domain kustom yang dioptimalkan tepi.
5. Untuk konfigurasi Endpoint, pilih Edit.
6. Untuk sertifikat ACM, pilih sertifikat dari daftar dropdown.
7. Pilih Simpan perubahan untuk mulai memutar sertifikat untuk nama domain kustom.

REST API

Panggil [domainname:update action](#), yang menentukan ARN dari sertifikat ACM baru untuk nama domain yang ditentukan.

AWS CLI

Berikut ini [update-domain-name](#) memperbarui sertifikat ACM untuk nama domain yang dioptimalkan tepi.

```
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'"
```

Berikut ini [update-domain-name](#) memperbarui sertifikat ACM untuk nama domain Regional.

```
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'"
```

Panggil API Anda dengan nama domain khusus saat Anda menggunakan pemetaan jalur dasar

Memanggil API dengan nama domain khusus sama dengan memanggil API dengan nama domain defaultnya, asalkan URL yang benar digunakan.

Contoh berikut membandingkan dan membedakan satu set default URLs dan kustom yang sesuai URLs dari dua APIs (udxjefdanqf3duz) di Region tertentu (us-east-1), dan nama domain kustom yang diberikan (api.example.com).

ID API	Stage	URL default	Jalur dasar	URL kustom
udxjef	prod	https://udxjef.execute-api.us-east-1.amazonaws.com/prod	/toko hewan peliharaan	https://api.example.com/toko hewan peliharaan
udxjef	tst	https://udxjef.execute-api.us-east-1	/petdepot	https://api.example.com/petdepot

ID API	Stage	URL default	Jalur dasar	URL kustom
		.amazonaws.com/tst		
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/ toko buku	https://api.example.com/toko buku
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/rak buku	https://api.example.com/rak buku

Untuk fleksibilitas lebih lanjut tentang cara Anda mengarahkan lalu lintas ke Anda APIs, Anda dapat membuat aturan perutean. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

API Gateway mendukung nama domain kustom untuk API dengan menggunakan [Server Name Indication \(SNI\)](#). Anda dapat memanggil API dengan nama domain khusus menggunakan browser atau pustaka klien yang mendukung SNI.

API Gateway memberlakukan SNI pada distribusi CloudFront. Untuk informasi tentang cara CloudFront menggunakan nama domain kustom, lihat [Amazon CloudFront Custom SSL](#).

Memigrasikan nama domain kustom ke jenis endpoint API yang berbeda di API Gateway

Anda dapat memigrasikan nama domain kustom antara titik akhir yang dioptimalkan tepi dan Regional. Anda tidak dapat memigrasikan nama domain kustom publik ke nama domain kustom pribadi. Pertama-tama Anda menambahkan jenis konfigurasi titik akhir baru ke `endpointConfiguration.types` daftar yang ada untuk nama domain kustom. Selanjutnya, Anda menyiapkan data DNS untuk mengarahkan nama domain kustom ke titik akhir yang baru disediakan. Terakhir, Anda menghapus titik akhir nama domain kustom yang sudah usang.

Pertimbangan-pertimbangan

Berikut ini adalah pertimbangan untuk memigrasikan domain kustom Anda antara titik akhir API Regional dan titik akhir API yang dioptimalkan tepi:

- Nama domain kustom yang dioptimalkan tepi memerlukan sertifikat yang disediakan oleh ACM dari Wilayah AS Timur (Virginia N.). us-east-1 Sertifikat ini didistribusikan ke semua lokasi geografis.
- Nama domain kustom Regional memerlukan sertifikat yang disediakan oleh ACM di Wilayah yang sama yang menghosting API. Anda dapat memigrasikan nama domain kustom yang dioptimalkan tepi yang tidak ada di us-east-1 Wilayah ke nama domain kustom Regional dengan meminta sertifikat ACM baru dari Wilayah yang lokal ke API.
- Mungkin diperlukan waktu hingga 60 detik untuk menyelesaikan migrasi antara nama domain kustom yang dioptimalkan tepi dan nama domain kustom Regional. Waktu migrasi juga tergantung pada saat Anda memperbarui catatan DNS Anda.
- Anda hanya dapat menambahkan konfigurasi titik akhir tambahan jika mode akses titik akhir diatur ke. BASIC Setelah Anda memiliki dua konfigurasi titik akhir, Anda tidak dapat mengubah mode akses titik akhir. Untuk informasi selengkapnya, lihat [the section called “Mode akses titik akhir”](#).
- Jika nama domain kustom Anda menggunakan kebijakan keamanan yang dimulai denganSecurityPolicy_, ketika Anda menambahkan jenis konfigurasi titik akhir baru, mode akses titik akhir akan sama di kedua jenis titik akhir, dan Anda harus memilih kebijakan keamanan yang dimulai dengan SecurityPolicy_ untuk jenis konfigurasi titik akhir yang baru.

Migrasi nama domain kustom

Note

Untuk menyelesaikan migrasi, pastikan Anda menghapus titik akhir usang dari nama domain kustom Anda.

Prosedur berikut menunjukkan cara memigrasikan nama domain kustom yang dioptimalkan tepi ke nama domain kustom Regional.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.

3. Pilih nama domain kustom yang dioptimalkan tepi.
4. Untuk konfigurasi Endpoint, pilih Edit.
5. Pilih Tambahkan titik akhir Regional.
6. Untuk sertifikat ACM, pilih sertifikat.

Sertifikat Regional harus berada di Wilayah yang sama dengan API Regional.

7. Pilih Simpan perubahan.
8. Siapkan data DNS untuk mengarahkan nama domain kustom Regional ke nama host Regional ini. Untuk informasi selengkapnya, lihat [mengonfigurasi Route 53 untuk merutekan lalu lintas ke API Gateway](#).
9. Setelah mengonfirmasi bahwa konfigurasi DNS Anda menggunakan titik akhir yang benar, Anda menghapus konfigurasi titik akhir yang dioptimalkan tepi. Pilih nama domain kustom Anda, lalu untuk konfigurasi titik akhir yang dioptimalkan di tepi, pilih Hapus.
10. Konfirmasikan pilihan Anda dan hapus titik akhir.

AWS CLI

[update-domain-name](#) Perintah berikut memigrasikan nama domain kustom edge-optimized ke nama domain kustom Regional:

```
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" }
  ]'
```

Sertifikat Regional harus dari Wilayah yang sama dengan API Regional.

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}
```

Untuk nama domain kustom Regional yang dimigrasi, `regionalDomainName` properti yang dihasilkan menampilkan nama host API Regional. Anda harus menyiapkan catatan DNS untuk mengarahkan nama domain kustom Regional ke nama host Regional ini. Hal ini memungkinkan lalu lintas yang terikat pada nama domain kustom untuk diarahkan ke host Regional.

Setelah catatan DNS disetel, Anda dapat menghapus nama domain kustom yang dioptimalkan tepi. [update-domain-name](#) Perintah berikut menghapus nama domain kustom yang dioptimalkan tepi:

```
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"}
  ]'
```

Prosedur berikut menunjukkan cara memigrasikan nama domain kustom yang dioptimalkan tepi yang menggunakan kebijakan keamanan yang disempurnakan ke nama domain kustom Regional yang juga menggunakan kebijakan keamanan yang ditingkatkan.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain kustom yang dioptimalkan tepi.
4. Untuk konfigurasi Endpoint, pilih Edit.
5. Pilih Tambahkan titik akhir Regional.

6. Untuk sertifikat ACM, pilih sertifikat.

Sertifikat Regional harus berada di Wilayah yang sama dengan API Regional.

7. Untuk kebijakan Keamanan, pilih kebijakan keamanan yang dimulai dengan `SecurityPolicy_`.
8. Untuk mode akses Endpoint, pilih Basic.
9. Pilih Simpan perubahan.
10. Siapkan data DNS untuk mengarahkan nama domain kustom Regional ke nama host Regional ini. Untuk informasi selengkapnya, lihat [mengonfigurasi Route 53 untuk merutekan lalu lintas ke API Gateway](#).
11. Setelah mengonfirmasi bahwa konfigurasi DNS Anda menggunakan titik akhir yang benar, Anda menghapus konfigurasi titik akhir yang dioptimalkan tepi. Pilih nama domain kustom Anda, lalu untuk konfigurasi titik akhir yang dioptimalkan di tepi, pilih Hapus.
12. Konfirmasikan pilihan Anda dan hapus titik akhir.

AWS CLI

[update-domain-name](#) Perintah berikut memigrasikan nama domain kustom edge-optimized ke nama domain kustom Regional:

```
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" }
  ]'
```

Sertifikat Regional harus dari Wilayah yang sama dengan API Regional.

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}

```

Untuk nama domain kustom Regional yang dimigrasi, `regionalDomainName` properti yang dihasilkan menampilkan nama host API Regional. Anda harus menyiapkan catatan DNS untuk mengarahkan nama domain kustom Regional ke nama host Regional ini. Hal ini memungkinkan lalu lintas yang terikat pada nama domain kustom untuk diarahkan ke host Regional.

Setelah catatan DNS disetel, Anda dapat menghapus nama domain kustom yang dioptimalkan tepi. [update-domain-name](#) Perintah berikut menghapus nama domain kustom yang dioptimalkan tepi:

```

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"}
  ]'

```

Prosedur berikut menunjukkan cara memigrasikan nama domain kustom Regional ke nama domain kustom yang dioptimalkan tepi.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom Regional.

4. Untuk konfigurasi Endpoint, pilih Edit.
5. Pilih Tambahkan titik akhir yang dioptimalkan tepi.
6. Untuk sertifikat ACM, pilih sertifikat.

Sertifikat domain yang dioptimalkan tepi harus dibuat di Wilayah. us-east-1

7. Pilih Simpan.
8. Siapkan data DNS untuk mengarahkan nama domain kustom yang dioptimalkan tepi ke nama host yang dioptimalkan tepi ini. Untuk informasi selengkapnya, lihat [mengonfigurasi Route 53 untuk merutekan lalu lintas ke API Gateway](#).
9. Setelah mengonfirmasi bahwa konfigurasi DNS Anda menggunakan titik akhir yang benar, Anda menghapus konfigurasi titik akhir Regional. Pilih nama domain kustom Anda, dan kemudian untuk konfigurasi titik akhir Regional, pilih Hapus.
10. Konfirmasikan pilihan Anda dan hapus titik akhir.

AWS CLI

[update-domain-name](#) Perintah berikut memigrasikan nama domain kustom Regional Anda ke nama domain kustom yang dioptimalkan tepi:

```
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" }
  ]'
```

Sertifikat domain yang dioptimalkan tepi harus dibuat di Wilayah. us-east-1

Outputnya akan terlihat seperti berikut:

```
{
  "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"
  }

```

Untuk nama domain kustom yang ditentukan, API Gateway menampilkan nama host API yang dioptimalkan tepi sebagai nilai properti. `distributionDomainName` Anda harus menetapkan catatan DNS untuk mengarahkan nama domain kustom yang dioptimalkan tepi ke nama domain distribusi ini. Ini memungkinkan lalu lintas yang terikat pada nama domain kustom yang dioptimalkan tepi untuk dirutekan ke nama host API yang dioptimalkan tepi.

Setelah catatan DNS ditetapkan, Anda dapat menghapus jenis REGION titik akhir dari nama domain kustom. [update-domain-name](#) Perintah berikut menghapus tipe endpoint Regional:

```

aws apigateway update-domain-name \
  --domain-name api.example.com \
  --patch-operations '[
    {"op":"remove", "path":"/endpointConfiguration/types", "value":"REGIONAL"},
    {"op":"remove", "path":"regionalCertificateArn"}
  ]'

```

Outputnya terlihat seperti berikut:

```

{
  "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"
  }
}

```

Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway

Ketika Anda mengkonfigurasi mode routing untuk nama domain kustom Anda, Anda mengatur bagaimana lalu lintas masuk diarahkan ke Anda. APIs Anda mengirim lalu lintas ke APIs menggunakan aturan perutean, pemetaan API, atau aturan perutean dan pemetaan API. Bagian berikut menjelaskan kapan harus menggunakan aturan perutean, kapan menggunakan pemetaan API, dan cara mengatur mode perutean untuk nama domain kustom Anda.

Kapan menggunakan aturan perutean

Saat Anda menggunakan aturan perutean, Anda mengarahkan permintaan masuk yang cocok dengan kondisi tertentu ke tahapan REST APIs tertentu. Misalnya, aturan dapat merutekan permintaan ke `production` tahap `users` REST API Anda jika berisi header `version:v1` dan jalur dasar/`users`. Gunakan aturan perutean untuk membuat topologi perutean dinamis lanjutan yang mendukung kasus penggunaan seperti A/B pengujian atau peningkatan penggunaan versi baru Anda. APIs

Sebaiknya saat mengarahkan lalu lintas ke REST API, Anda menggunakan aturan perutean untuk nama domain kustom Anda. Anda dapat membuat ulang pemetaan API apa pun dengan menggunakan aturan perutean. Untuk informasi selengkapnya, lihat [the section called “Membuat ulang pemetaan API menggunakan aturan perutean”](#).

Untuk REST APIs, Anda juga dapat menggunakan aturan routing dan pemetaan API bersama-sama. Saat Anda menggunakan aturan perutean dan pemetaan API bersama-sama, API Gateway selalu mengevaluasi aturan perutean sebelum mengevaluasi pemetaan API apa pun. Gunakan aturan perutean dan pemetaan API bersama-sama untuk memigrasikan nama domain kustom Anda saat ini atau untuk menjelajahi aturan perutean.

Pertimbangan untuk aturan perutean

Pertimbangan berikut dapat memengaruhi penggunaan aturan perutean Anda:

- WebSocket atau HTTP APIs tidak didukung sebagai target APIs untuk aturan perutean.
- Jika nama domain kustom Anda memiliki pemetaan API ke REST dan HTTP APIs, aturan perutean tidak didukung.
- Anda dapat membuat aturan perutean untuk domain kustom pribadi ke REST API pribadi. Anda dapat membuat aturan perutean untuk domain kustom publik ke API Regional atau yang dioptimalkan tepi.

- Anda tidak dapat membuat aturan perutean untuk domain kustom publik ke API pribadi. Anda tidak dapat membuat aturan perutean untuk nama domain kustom pribadi ke API publik.

Pilih antara aturan perutean dan pemetaan API

Kami menyarankan bahwa jika memungkinkan, Anda menggunakan aturan routing. Hanya gunakan pemetaan API untuk mengirim lalu lintas ke HTTP atau WebSocket API.

Mengatur mode routing untuk nama domain kustom Anda

Anda dapat memilih mode perutean yang digunakan API Gateway untuk merutekan lalu lintas ke Anda APIs. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#). Bagian ini membahas mode routing untuk nama domain kustom. Anda harus mengatur mode perutean untuk nama domain kustom Anda untuk mengarahkan lalu lintas ke Anda APIs. Mode perutean berikut didukung:

- **ROUTING_RULE_THEN_API_MAPPING** — Gunakan mode ini untuk mengirim lalu lintas ke Anda APIs dengan aturan perutean dan pemetaan API. Dalam mode ini, semua aturan perutean diprioritaskan daripada pemetaan API apa pun. Untuk contoh mode ini, lihat [the section called “Contoh 2: Aturan perutean dan pemetaan API”](#).
- **ROUTING_RULE_ONLY** — Gunakan mode ini untuk hanya mengizinkan aturan routing untuk mengirim lalu lintas ke Anda. APIs Ketika nama domain kustom Anda menggunakan mode ini, Anda tidak dapat membuat pemetaan API, tetapi Anda dapat menggunakan [get-api-mappings](#) perintah untuk melihatnya. Pemanggil API tidak dapat menggunakan pemetaan API untuk mengakses nama domain ini.
- **API_MAPPING_ONLY** — Gunakan mode ini untuk hanya mengizinkan pemetaan API untuk mengirim lalu lintas ke Anda. APIs Saat nama domain kustom Anda menggunakan mode ini, Anda tidak dapat membuat aturan perutean, tetapi Anda dapat menggunakan `list-routing-rules` perintah untuk melihatnya. Pemanggil API tidak dapat menggunakan aturan perutean untuk mengakses nama domain ini.

Ini adalah mode routing default untuk semua nama domain yang ada, dan nama domain baru yang Anda buat.

Ketika Anda membuat nama domain kustom menggunakan `apigateway`, **API_MAPPING_ONLY** dipanggil **BASE_PATH_MAPPING_ONLY** dan **ROUTING_RULE_THEN_API_MAPPING**

dipanggil `ROUTING_RULE_THEN_BASE_PATH_MAPPING`. Perilaku ini hanya ada untuk AWS CLI, CloudFormation, atau apa pun SDKs, tidak di Konsol Manajemen AWS.

Prosedur berikut menunjukkan cara mengubah mode routing untuk nama domain kustom yang ada. Saat Anda mengubah mode perutean nama domain kustom Anda, pemanggil API tidak dapat mengakses nama domain Anda menggunakan mode perutean yang tidak didukung.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain khusus.
4. Untuk detail Domain, pilih Edit.
5. Untuk mode Routing, pilih `API_MAPPINGROUTING_RULE_THEN_`.
6. Pilih Simpan.

Jika Anda mengubah mode perutean ke `ROUTING_RULE_ONLY` atau `API_MAPPING_ONLY`, pemetaan API atau aturan perutean apa pun yang Anda buat akan dihapus dari halaman detail nama domain konsol. Jika Anda mengubah mode perutean untuk mendukung aturan perutean atau pemetaan API, sumber daya ini akan kembali.

AWS CLI - apigatewayv2

[update-domain-name](#) Perintah berikut memperbarui nama domain untuk menggunakan mode `ROUTING_RULE_THEN_API_MAPPING` routing:

```
aws apigatewayv2 update-domain-name \  
  --domain-name 'api.example.com' \  
  --routing-mode "ROUTING_RULE_THEN_API_MAPPING"
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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

[update-domain-name](#) Perintah berikut memperbarui nama domain kustom pribadi untuk menggunakan mode ROUTING_RULE_THEN_BASE_PATH_MAPPING perutean:

```

aws apigateway update-domain-name \
  --domain-name 'private.example.com' \
  --patch-operations "op='replace',path='/
routingMode',value='ROUTING_RULE_THEN_BASE_PATH_MAPPING'"

```

Outputnya akan terlihat seperti berikut:

```

{
  "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"
}

```

```
}
```

Aturan perutean untuk menghubungkan tahapan API ke nama domain khusus untuk REST APIs

Aturan routing adalah serangkaian kondisi yang ketika dicocokkan, memanggil tindakan.

Misalnya, aturan dapat merutekan permintaan masuk ke nama domain kustom yang berisi header `Hello:World` dan berisi jalur dasar `users` ke `production` tahap REST API.

Aturan dievaluasi dalam urutan prioritas, dan jika Anda menyetel mode perutean `ROUTING_RULE_THEN_API_MAPPING`, API Gateway selalu mengevaluasi semua aturan perutean sebelum mengevaluasi pemetaan API apa pun. Daftar berikut menjelaskan bagaimana aturan routing menggunakan kondisi, tindakan, dan prioritas.

Ketentuan

Bila syarat untuk suatu peraturan terpenuhi, maka tindakannya dilakukan. API Gateway mendukung hingga dua kondisi header dan satu kondisi jalur. API Gateway mengevaluasi kondisi header dan kondisi jalur dasar secara bersamaan.

Anda dapat membuat aturan tanpa syarat apa pun. Saat API Gateway mengevaluasi aturan ini, tindakan selalu dilakukan. Anda dapat membuat aturan tanpa syarat apa pun sebagai aturan `catch-all`.

Untuk informasi selengkapnya tentang kondisi header, lihat [the section called “Ketentuan header pertandingan”](#). Untuk informasi selengkapnya tentang kondisi jalur, lihat [the section called “Cocokkan kondisi jalur dasar”](#).

Tindakan

Tindakan adalah hasil dari kondisi pencocokan dengan aturan routing. Saat ini, satu-satunya tindakan yang didukung adalah memanggil tahap REST API.

Setiap aturan dapat memiliki satu tindakan.

Prioritas

Prioritas menentukan urutan aturan yang dievaluasi, dari nilai terendah hingga nilai tertinggi. Aturan tidak dapat memiliki prioritas yang sama.

Anda dapat menetapkan prioritas dari 1-1.000.000. Jika aturan memiliki prioritas satu, API Gateway mengevaluasinya terlebih dahulu. Kami menyarankan bahwa ketika Anda membuat

aturan, Anda menambahkan kesenjangan dalam prioritas. Ini membantu Anda mengubah prioritas aturan dan menambahkan aturan baru. Untuk informasi selengkapnya, lihat [the section called “Mengubah prioritas aturan routing”](#).

Untuk contoh cara API Gateway mengevaluasi aturan perutean, lihat [the section called “Contoh bagaimana API Gateway mengevaluasi aturan perutean”](#)

Jenis kondisi aturan perutean API Gateway

Bagian berikut menjelaskan jenis kondisi aturan routing. API Gateway hanya cocok dengan aturan jika semua kondisi benar.

Ketentuan header pertandingan

Saat Anda membuat kondisi header, Anda dapat mencocokkan nama header dan nilai glob header, seperti `Hello:World`. API Gateway menggunakan kecocokan literal untuk memvalidasi kondisi header kecocokan. Kondisi Anda dapat menggunakan hingga dua header yang digunakan di AND antara mereka. Misalnya, kondisi Anda dapat cocok jika permintaan masuk berisi `Hello:World` dan `x-version:beta`.

Pencocokan nama header tidak peka huruf besar/kecil, tetapi nilai glob header peka huruf besar/kecil. `Hello:World` akan cocok `hello:World`, tapi tidak `Hello:world`.

Untuk daftar nilai header terbatas lihat [the section called “Pembatasan”](#).

Menggunakan wildcard dengan kondisi header

Anda hanya dapat menggunakan wildcard dalam nilai header glob, dan wildcard harus `*prefix-match`, `suffix-match` atau `*contains*`. Tabel berikut menunjukkan contoh cara menggunakan wildcard untuk pencocokan kondisi header.

Kondisi header	Permintaan yang cocok dengan aturan perutean	Permintaan yang tidak cocok dengan aturan perutean
<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>	<code>x-version: alpha</code>	<code>x-version: account</code>

Kondisi header	Permintaan yang cocok dengan aturan perutean	Permintaan yang tidak cocok dengan aturan perutean
	• <code>x-version: beta</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> dan <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> dan <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>	Tidak ada

Jika Anda membuat kondisi untuk beberapa nilai header, seperti `Accept: application/json, text/xml`, kami sarankan Anda menggunakan `*contains*` untuk kondisi header Anda dan menghindari membuat kondisi menggunakan karakter koma (,).

Karena API Gateway cocok dengan kondisi header secara harfiah, kecocokan semantik mungkin dirutekan secara berbeda. Tabel berikut menunjukkan perbedaan dalam hasil aturan routing.

Kondisi header	Permintaan yang cocok dengan aturan perutean	Permintaan yang tidak cocok dengan aturan perutean
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	Tidak ada

Cocokkan kondisi jalur dasar

Saat Anda membuat kondisi jalur dasar, jika permintaan masuk berisi jalur yang Anda tentukan, aturannya cocok. Pencocokannya peka huruf besar/kecil, jadi jalurnya tidak New/Users akan cocoknew/users.

Anda dapat membuat kondisi jalur dasar hanya untuk satu jalur dasar.

Untuk daftar kondisi jalur dasar terbatas, [the section called “Pembatasan”](#).

Lepaskan jalur dasar dengan kondisi jalur dasar

Saat Anda membuat kondisi jalur dasar, Anda dapat memilih untuk menghapus jalur dasar. Saat Anda menghapus jalur dasar, API Gateway menghapus jalur dasar yang cocok yang masuk saat memanggil API target. Ini adalah perilaku yang sama seperti ketika Anda menggunakan pemetaan API. Jika Anda tidak menghapus jalur dasar, API Gateway meneruskan seluruh jalur dasar ke API target. Sebaiknya Anda hanya menghapus jalur dasar saat membuat ulang pemetaan API.

Tabel berikut menunjukkan contoh bagaimana API Gateway mengevaluasi kondisi jalur dasar strip.

Kondisi	Jalur dasar strip	Permintaan masuk	Hasil
Jika jalur dasar berisi PetStoreShopper/dogs	True	GET https://example.com/PetStoreShopper/dogs	API Gateway memanggil GET metode / sumber daya.

Kondisi	Jalur dasar strip	Permintaan masuk	Hasil
Jika jalur dasar berisi PetStoreShopper/dogs .	False	GET https://example.com/PetStoreShopper/dogs	API Gateway memanggil GET metode PetStoreShopper/dogs sumber daya.
Jika jalur dasar berisi PetStoreShopper	True	GET https://example.com/PetStoreShopper/dogs	API Gateway memanggil GET metode dogs sumber daya.
Jika jalur dasar berisi PetStoreShopper	False	GET https://example.com/PetStoreShopper/dogs	API Gateway memanggil GET metode PetStoreShopper/dogs sumber daya.
Jika jalur dasar berisi PetStoreShopper	True	GET https://example.com/PetStoreShopper?birds=available	API Gateway memanggil GET metode / sumber daya dengan parameter string kueribirds=available .
Jika jalur dasar berisi PetStoreShopper	False	GET https://example.com/PetStoreShopper?birds=available	API Gateway memanggil GET metode /PetStoreShopper sumber daya dengan parameter string kueribirds=available .

Pembatasan

- API target dan nama domain khusus harus berada di AWS akun yang sama.
- Setiap aturan dapat memiliki satu API target.
- Anda hanya dapat membuat aturan perutean untuk nama domain kustom pribadi ke API pribadi, dan untuk nama domain kustom publik ke API publik. Anda tidak dapat mencampur sumber daya publik dan swasta.
- Jika nama domain kustom Anda memiliki pemetaan API ke REST dan HTTP APIs, aturan perutean tidak didukung.
- Jumlah prioritas maksimum adalah 1.000.000.
- Pembatasan header:
 - Setiap anyOf kondisi hanya dapat berisi satu nilai header.
 - Satu-satunya karakter yang diizinkan untuk nama header dan nilai glob header ditentukan oleh [RFC 7230](#), yaitu, a-z, A-Z 0-9, dan karakter khusus berikut: *? - ! # \$ % & ' . ^ _ ` | ~
 - Anda dapat menggunakan wildcard dalam nilai header glob, tetapi wildcard harus *prefix-match,, suffix-match* atau. *contains* Anda tidak dapat menggunakan * di tengah nilai glob header.
 - Nama header wildcard tidak didukung.
 - Nama header harus kurang dari 40 karakter.
 - Nilai header glob harus kurang dari 128 karakter.
 - Nilai header glob untuk kecocokan infix harus kurang dari 40 karakter.
 - Header berikut tidak didukung sebagai kondisi:
 - access-control-*
 - apigw-*
 - Authorization
 - Connection
 - Content-Encoding
 - Content-Length
 - Content-Location
 - Forwarded
 - Keep-Alive

- 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
- Batasan jalur dasar:
 - Panjang jalur dasar harus kurang dari 128 karakter.
 - Jalur dasar harus berisi hanya huruf, angka, dan karakter berikut: `$-_.+!*'()/`.

Karakter ini tidak didukung untuk ekspresi reguler (regex).

 - Jalur dasar tidak dapat memulai atau mengakhiri dengan karakter garis miring terbalik (`\`).

Contoh bagaimana API Gateway mengevaluasi aturan perutean

Bagian berikut menunjukkan empat contoh bagaimana API Gateway mengevaluasi aturan routing dan pemetaan API.

Contoh 1: Aturan perutean saja

Dalam contoh ini, nama domain khusus `https://petstore.example.com` memiliki mode perutean yang disetel ke `ROUTING_RULE_ONLY` dan memiliki aturan dan prioritas perutean berikut.

ID aturan	Prioritas	Ketentuan	Tindakan
abc123	10	Jika permintaan berisi header: Hello:World	Target API 1
zzz000	50	Jika permintaan berisi header: Accept:image/webp dan Pet:Dog-* dan jika jalur dasar berisi PetStoreShopper	Target API 2
efg456	100	Tidak ada	Target API 3

Tabel berikut menunjukkan bagaimana API Gateway menerapkan aturan perutean sebelumnya ke permintaan contoh.

Permintaan	API yang dipilih	Penjelasan
<code>https://petstore.example.com -h "Hello:World"</code>	Target API 1	Permintaan cocok dengan aturan abc123 routing.
<code>https://petstore.example.com/PetStoreShopper -h "Hello:World", "Pet:Dog-Bella", "Accept:image/webp"</code>	Target API 1	API Gateway mengevaluasi semua aturan perutean dalam urutan prioritas. Aturan perutean abc123 memiliki prioritas pertama dan kondisinya cocok, sehingga API Gateway memanggil Target API 1. Meskipun kondisi permintaan juga cocok dengan aturan perutean zzz000, API Gateway tidak mengevaluasi

Permintaan	API yang dipilih	Penjelasan
		asi aturan perutean lainnya setelah membuat kecocokan.
<code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella", "Accept:image/webp"</code>	Target API 2	Permintaan cocok dengan aturan zzz000 routing. Ini adalah pertandingan karena itu Pet:Dog-Bella adalah string yang cocok Pet:Dog-*
<code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella"</code>	Target API 3	Permintaan tidak cocok dengan aturan abc123 routing. Permintaan tidak cocok dengan aturan perutean zzz000 karena semua header yang diperlukan tidak ada. Aturan prioritas berikutnya cocok dengan semua permintaan yang masuk, jadi API Gateway memanggil Target API 3.

Contoh 2: Aturan perutean dan pemetaan API

Dalam contoh ini, nama domain khusus `https://petstore.diagram.example.com` memiliki mode perutean yang disetel ke `ROUTING_RULE_THEN_API_MAPPING` dan memiliki aturan perutean dan pemetaan API berikut.

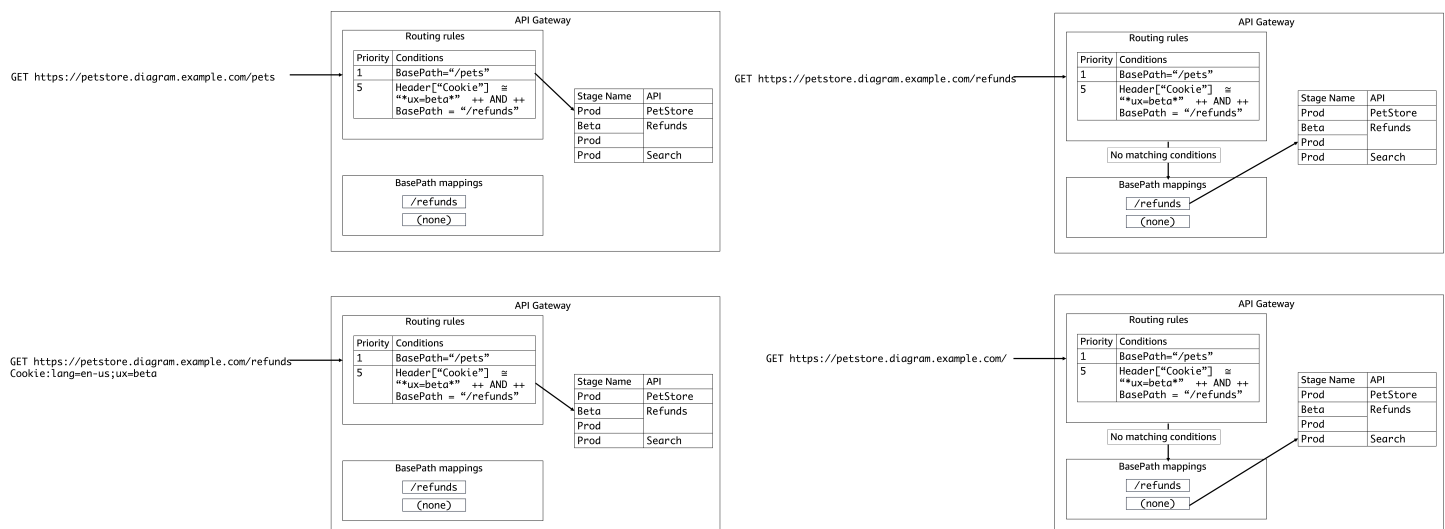
ID aturan	Prioritas	Ketentuan	Tindakan
abc123	1	Jika permintaan basis berisi pets	Memanggil Prod tahap PetStore API.
000zzz	5	Jika permintaan berisi header:Cookie:*ux=beta* dan dan	Memanggil Beta tahap Refunds API.

ID aturan	Prioritas	Ketentuan	Tindakan
		jika jalur dasar berisi /refunds	

Tabel berikut menunjukkan pemetaan API untuk. <https://petstore.backup.example.com>

Pemetaan API	API yang dipilih
/refunds	Memanggil Prod tahap Refunds API.
(none)	Memanggil Prod tahap Search API.

Diagram berikut menunjukkan bagaimana API Gateway menerapkan aturan perutean sebelumnya dan pemetaan API ke permintaan contoh. Contoh permintaan dirangkum dalam tabel setelah diagram ini.



Tabel berikut menunjukkan bagaimana API Gateway menerapkan aturan perutean sebelumnya dan pemetaan API ke permintaan contoh.

Permintaan	API yang dipilih	Penjelasan
https://petstore.backup.example.com/pets	ProdTahap PetStore API.	Permintaan cocok dengan aturan abc123 perutean.

Permintaan	API yang dipilih	Penjelasan
<code>https://petstore.d iagram.example.com /refunds -h "Cookie:l ang=en-us;ux=beta"</code>	BetaTahap Refunds API.	Permintaan cocok dengan aturan 000zzz perutean. CookieHeader berisi *contains* kecocokan yang benar dan pencocokan jalur dasar untuk kondisi ini.
<code>https://petstore.d iagram.example.com/ refunds</code>	ProdTahap Refunds API.	Permintaan tidak memiliki header yang diperluka n untuk mencocokkan aturan perutean. zzz000 Jika API Gateway tidak berhasil mencocokkan aturan perutean, itu akan kembali ke pemetaan API. API Gateway dapat memetakan jalur dasar ke Prod tahap Refunds API.
<code>https://petstore.d iagram.example.com/</code>	ProdTahap Search API.	Permintaan mencocokkan pemetaan API dengan jalur (none) kosong.

Contoh 3: Aturan perutean dan pemetaan API dengan beberapa level

Dalam contoh ini, nama domain khusus `https://petstore.backup.example.com` memiliki mode perutean yang disetel ke `ROUTING_RULE_THEN_API_MAPPING` dan memiliki aturan perutean dan pemetaan API berikut.

Tabel berikut menunjukkan aturan routing untuk `https://petstore.backup.example.com`.

ID aturan	Prioritas	Ketentuan	Tindakan
abc123	10	Jika permintaan berisi header: Hello:World	Target API 1

ID aturan	Prioritas	Ketentuan	Tindakan
000zzz	50	Jika permintaan berisi header:Accept:image/webp dan Pet:Dog-* dan jika jalur dasar berisi PetStoreShopper	Target API 2

Tabel berikut menunjukkan pemetaan API untuk. `https://petstore.backup.example.com`

Pemetaan API	API yang dipilih
PetStoreShopper	Target API 3
PetStoreShopper/cats	Target API 4

Tabel berikut menunjukkan bagaimana API Gateway menerapkan aturan perutean sebelumnya dan pemetaan API ke permintaan contoh.

Permintaan	API yang dipilih	Penjelasan
<code>https://petstore.example.com/PetStoreShopper -h "Accept:image/webp", "Pet:Cats"</code>	Target API 3	Permintaan tidak memiliki header yang diperlukan untuk mencocokkan aturan perutean. zzz000 Jika API Gateway tidak berhasil mencocokkan aturan perutean, itu akan kembali ke pemetaan API. API Gateway dapat memetakan jalur dasar ke Target API 3.
<code>https://petstore.example.com/PetStore</code>	Target API 1	Permintaan cocok dengan aturan abc123 perutean. Jika mode perutean disetel

Permintaan	API yang dipilih	Penjelasan
eShopper/cats -h "Hello:World"		keROUTING_RULE_THEN_API_MAPPING , aturan perutean selalu diprioritaskan daripada pemetaan API.
https://petstore.example.com/Admin -h "Pet:Dog-Bella"	Tidak ada	Permintaan tidak cocok dengan aturan perutean atau pemetaan API apa pun. Karena tidak ada aturan perutean default, API Gateway menolak panggilan dan mengirimkan kode status kepada 403 Forbidden pemanggil.

Contoh 4: Aturan perutean untuk nama domain wildcard

Dalam contoh ini, nama domain kustom `https://*.example.com` adalah nama domain wildcard. Wildcard mendukung semua subdomain yang merutekan kembali ke domain yang sama. Contoh aturan perutean berikut mengubah perilaku ini untuk memungkinkan subdomain merutekan ke target yang berbeda APIs dengan menggunakan header. Host

Tabel berikut menunjukkan aturan routing untuk `https://*.example.com`.

ID aturan	Prioritas	Ketentuan	Tindakan
abc123	10	Jika permintaan berisi header: Host:a.example.com	Target API 1
000zzz	50	Jika permintaan berisi header: Host:b.example.com	Target API 2
efg456	500	Tidak ada	Target API 3

Tabel berikut menunjukkan bagaimana API Gateway menerapkan aturan perutean sebelumnya ke permintaan contoh.

Permintaan	API yang dipilih	Penjelasan
<code>https://a.example.com</code>	Target API 1	HostHeader adalah <code>.example.com</code> . Permintaan ini cocok dengan aturan <code>abc123</code> routing.
<code>https://b.example.com</code>	Target API 2	HostHeader adalah <code>b.example.com</code> . Permintaan ini cocok dengan aturan <code>000zzz</code> routing.
<code>https://testing.example.com</code>	Target API 3	Ini cocok dengan aturan perutean catch-all. <code>efg456</code>

Cara menggunakan aturan routing

Anda dapat membuat aturan perutean menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS SDK apa pun. Setelah Anda membuat aturan, Anda dapat mengubah prioritasnya.

Buat aturan perutean

Prosedur berikut menunjukkan cara membuat aturan routing untuk nama domain kustom dengan mode routing diatur ke salah satu `at Routing Rule THEN API Mapping` atau `Routing Rule Only`.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain khusus.
4. Pada tab Routing details, pilih Add routing rule.
5. Pilih Tambahkan kondisi baru untuk menambahkan kondisi baru.

Anda dapat menambahkan header atau kondisi jalur dasar. Untuk mencocokkan semua permintaan yang masuk dengan nama domain kustom Anda, jangan tambahkan kondisi.

6. Untuk Action, gunakan dropdown untuk memilih API target dan tahap target Anda.
7. Pilih Berikutnya.
8. Di bidang prioritas, masukkan nomor untuk prioritas Anda.

API Gateway mengevaluasi aturan dalam urutan prioritas, dari nilai terendah hingga nilai tertinggi.

Jika Anda membuat aturan tanpa syarat, kami sarankan Anda menggunakan prioritas nilai tinggi.

9. Pilih Buat aturan perutean.

AWS CLI

[create-routing-rule](#) Perintah berikut membuat aturan routing dengan prioritas 50. Dalam contoh ini, API Gateway merutekan permintaan masuk yang memiliki header `Hello:World` dan `x-version:beta` dan jalur dasar `PetStoreShopper` ke API target. `a1b2c3`

```
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"
  }
}
]'
```

Outputnya akan terlihat seperti berikut ini.

```

{
  "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"
}
```

Mengubah prioritas aturan routing

Anda dapat mengubah prioritas aturan routing. Ini segera berlaku dan mungkin memengaruhi cara konsumen API memanggil nama domain kustom Anda. Kami menyarankan bahwa ketika Anda menetapkan prioritas aturan routing Anda, Anda meninggalkan kesenjangan di antara aturan.

Misalnya, pertimbangkan dua aturan routing, aturan abc123 dengan prioritas 50 dan aturan zzz000 dengan prioritas 150. Untuk mengubah prioritas aturan agar API Gateway mengevaluasi aturan zzz000 terlebih dahulu, Anda dapat mengubah prioritas aturan zzz000 menjadi 30.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain khusus.
4. Pada tab Routing details, pilih aturan routing Anda, lalu pilih Edit.

5. Pilih Berikutnya.
6. Untuk prioritas, masukkan prioritas baru.
7. Pilih Simpan perubahan.

AWS CLI

[put-routing-rule](#) Perintah berikut mengubah prioritas aturan abc123 routing.

```
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"
  }
}
```

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}
```

Membuat ulang pemetaan API menggunakan aturan perutean

Anda dapat membuat ulang pemetaan API menggunakan aturan perutean. Untuk membuat ulang pemetaan API, pastikan untuk mengaktifkan striping jalur dasar. Ini mempertahankan perilaku pemetaan API. Untuk informasi selengkapnya, lihat [the section called “Lepaskan jalur dasar dengan kondisi jalur dasar”](#).

Tutorial berikut menunjukkan cara membuat ulang pemetaan API `https:// api.example.com/orders/v2/items/categories/5` sebagai aturan perutean dan cara memperbarui log akses Anda untuk mencatat aturan perutean ID API Gateway yang digunakan untuk mengirim lalu lintas ke API Anda.

Konsol Manajemen AWS

Untuk mengatur mode perutean ke `ROUTING_RULE_THEN_ API_MAPPING`

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain kustom Anda.
4. Untuk detail Domain, pilih Edit.
5. Untuk mode Routing, pilih `API_MAPPINGROUTING_RULE_THEN_`.
6. Pilih Simpan.

Setelah Anda mengatur mode routing, Anda membuat aturan routing.

Untuk membuat aturan routing

1. Pada tab Routing details, pilih Add routing rule.
2. Pilih Tambahkan kondisi baru dan kemudian pilih Path.
3. Untuk Path, masuk **orders/v2/items/categories/5**.
4. Untuk jalur dasar Strip, pilih Aktif.
5. Untuk API Target, pilih API target Anda.
6. Untuk tahap Target, pilih tahap target Anda.
7. Pilih Berikutnya.
8. Untuk prioritas, masukkan prioritas.

Bahkan jika Anda mempertahankan pemetaan API yang ada, API Gateway akan selalu menggunakan aturan perutean baru karena aturan perutean selalu diprioritaskan daripada pemetaan API.

9. Pilih Simpan perubahan.

Setelah Anda membuat aturan perutean, perbarui format log akses untuk tahap Anda atau buat log baru untuk mengonfirmasi bahwa API Gateway menggunakan aturan perutean Anda untuk mengirim lalu lintas ke API Anda.

Untuk memperbarui log akses Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda.
3. Di panel navigasi utama, pilih Tahapan.
4. Untuk Log dan penelusuran, pilih Edit.

Jika Anda tidak memiliki grup log, lihat [the section called “CloudWatch log”](#).

5. Tambahkan **\$context.customDomain.routingRuleIdMatched** ke format log Anda.

Grup log ini mencatat ID aturan perutean yang digunakan API Gateway untuk mengirim lalu lintas ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Saya tidak tahu bagaimana API Gateway mengirim lalu lintas ke saya APIs”](#).

6. Pilih Simpan.

Setelah memperbarui log akses, panggil nama domain kustom Anda. Berikut ini adalah contoh perintah curl untuk memanggil nama domain kustom `https://api.example.com` dengan jalur dasar. `orders/v2/items/categories/5`

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

Setelah Anda berhasil memanggil nama domain kustom Anda, konfirmasi bahwa CloudWatch Log menunjukkan `routingRuleIdMatched` Untuk mempelajari cara menggunakan konsol CloudWatch Log untuk melihat grup log, lihat [the section called “Lihat peristiwa log di CloudWatch konsol”](#).

AWS CLI

1. Gunakan [update-domain-name](#) perintah berikut untuk memperbarui nama domain `api.example.com` untuk menggunakan mode `ROUTING_RULE_THEN_API_MAPPING` routing.

```
aws apigatewayv2 update-domain-name \  
  --domain-name 'api.example.com' \  
  --routing-mode ROUTING_RULE_THEN_API_MAPPING
```

2. Gunakan [create-routing-rule](#) perintah berikut untuk membuat aturan routing baru untuk membuat ulang pemetaan API. `https://api.example.com/orders/v2/items/categories/5`

```
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
  }
}
```

- Gunakan perintah [update-stage](#) berikut untuk memperbarui format log akses untuk menyertakan variabel. `$context.customDomain.routingRuleIdMatched` Variabel ini mencatat ID aturan perutean yang digunakan API Gateway untuk mengirim lalu lintas ke API Anda. Anda menggunakan log ini untuk mengonfirmasi bahwa API Gateway menggunakan aturan perutean Anda untuk mengirim lalu lintas ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Saya tidak tahu bagaimana API Gateway mengirim lalu lintas ke saya 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'"
```

Jika Anda tidak memiliki grup log, lihat [the section called “CloudWatch log”](#).

- Gunakan contoh perintah curl berikut untuk memanggil nama domain kustom Anda dengan jalur dasar. `orders/v2/items/categories/5`

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

- Gunakan [filter-log-events](#) perintah berikut untuk mendapatkan peristiwa log dari grup log `access-log-group-orders` yang berisi ID `abc123` aturan routing.

```
aws logs filter-log-events --log-group-name access-log-group-orders --filter-
pattern abc123
```

Ini mengonfirmasi bahwa API Gateway menggunakan aturan perutean untuk mengirim lalu lintas ke API Anda.

Memecahkan masalah dengan aturan perutean

Panduan pemecahan masalah berikut dapat membantu menyelesaikan masalah dengan aturan perutean Anda.

Saya tidak tahu bagaimana API Gateway mengirim lalu lintas ke saya APIs

Anda dapat menggunakan log akses untuk tahap REST API untuk mencatat dan memecahkan masalah aturan perutean Anda. Anda dapat melihat ID aturan perutean yang digunakan API Gateway untuk mengirim lalu lintas ke API menggunakan `$context.customDomain.routingRuleIdMatched` variabel. Untuk melihat pemetaan API yang digunakan API Gateway untuk mengirim lalu lintas ke API Anda, gunakan `$context.customDomain.basePathMatched` variabel tersebut.

Untuk mencatat aturan perutean, Anda perlu mengonfigurasi ARN [peran CloudWatch Log yang sesuai](#) untuk akun Anda dan membuat grup log.

Contoh grup log akses berikut dapat mengambil informasi yang relevan untuk memecahkan masalah aturan perutean dan pemetaan API. API Gateway hanya mengisi variabel konteks untuk mekanisme perutean yang digunakannya, jika tidak, variabel konteksnya. -

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
```

Kami juga menyarankan Anda mengonfirmasi mode perutean untuk nama domain kustom Anda. Untuk informasi selengkapnya, lihat [the section called “Mengatur mode routing untuk nama domain kustom Anda”](#).

Saya tidak dapat mengaktifkan aturan perutean pada nama domain khusus saya

Anda mungkin menerima error berikut dari 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.
```

Anda akan menerima kesalahan ini jika memiliki atau memiliki kebijakan IAM yang menolak akses ke [BasePathMapping](#) atau [ApiMapping](#). Bila Anda mengaktifkan aturan perutean untuk nama domain kustom, meskipun kebijakan Anda akan terus menolak akses ke [BasePathMapping](#) atau [ApiMapping](#), kebijakan yang sama dapat digunakan untuk mengakses [RoutingRule](#). Ini mungkin memungkinkan pengguna untuk mengubah perilaku perutean nama domain kustom Anda.

Misalnya, jika Anda memiliki kebijakan seperti berikut:

```
{  
  "Sid": "DenyCreatingApiMappings",  
  "Effect": "Deny",  
  "Action": "apigateway:POST",  
  "Resource": [  
    "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"  
  ]  
}
```

Saat Anda mengaktifkan aturan perutean `example.com`, kebijakan ini akan terus menolak akses untuk membuat [ApiMapping](#) tetapi tidak akan menolak akses untuk membuat [RoutingRule](#) file.

Kami menyarankan Anda mengaudit kebijakan IAM di akun Anda. Contoh kebijakan berikut akan menolak akses untuk membuat `ApiMapping`, `BasePathMapping`, dan `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/*"
  ]
}
```

Setelah mengonfirmasi bahwa semua kebijakan telah diperbarui, Anda dapat memperbarui setelan tingkat akun API untuk mengaktifkan aturan perutean untuk Wilayah.

Gunakan perintah [update-account](#) berikut untuk memperbarui setelan setelan tingkat akun API Anda untuk Wilayah:

```
aws apigateway update-account --patch-operations 'op=remove,path=/
features,value=BlockedForRoutingRules' --region us-west-2
```

Setelah memperbarui setelan tingkat akun API, Anda dapat mengubah mode perutean nama domain kustom Anda. Anda juga dapat terus menggunakan kebijakan IAM untuk menolak akses ke `RoutingRules`, `ApiMapping` atau `BasePathMapping`.

Gunakan pemetaan API untuk menghubungkan tahapan API ke nama domain khusus untuk REST APIs

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain kustom. Ini mengirimkan lalu lintas ke Anda APIs melalui nama domain kustom Anda.

Pemetaan API menentukan API, tahap, dan jalur opsional yang akan digunakan untuk pemetaan. Misalnya, Anda dapat `https://api.example.com/orders` memetakan ke `production` tahap API.

Anda dapat memetakan tahap HTTP dan REST API ke nama domain kustom yang sama.

Sebelum membuat pemetaan API, Anda harus memiliki API, panggung, dan nama domain khusus. Untuk mempelajari selengkapnya tentang membuat nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Permintaan masuk ke nama domain kustom Anda

Saat Anda memetakan nama domain kustom ke tahap API Anda, API Gateway menghapus jalur dasar yang masuk. Ini menghapus jalur dasar yang dipetakan dari pemanggilan ke API. Misalnya, jika pemetaan jalur dasar Anda `https://api.example.com/orders/shop/5` ke test panggung, dan Anda menggunakan permintaan berikut `https://api.example.com/orders/shop/5/hats`, API Gateway akan memanggil `/hats` sumber daya test tahap API Anda, bukan sumber daya `orders/shop/5/hats`.

Memetakan permintaan API

Berikut ini menjelaskan bagaimana API Gateway mengevaluasi pemetaan API.

Anda dapat membuat pemetaan API menggunakan pemetaan tingkat tunggal, seperti pemetaan API dari `orders` beta tahap API dan pemetaan API dari `shipping` tahap API. alpha Untuk nama domain kustom Regional dengan kebijakan keamanan TLS 1.2, API Gateway mendukung pemetaan API multi-level. Anda dapat membuat pemetaan API dari `orders/v1/items` alpha tahap API dan `orders/v2/items` ke beta tahap API. Saat Anda membuat pemetaan dengan beberapa level, API Gateway mengirimkan permintaan ke pemetaan API yang memiliki jalur pencocokan terpanjang.

Anda dapat membuat pemetaan API ke jalur (none) kosong. Jika tidak ada jalur yang cocok dengan permintaan, API Gateway mengirimkan permintaan ke jalur kosong (none).

Dalam contoh ini, nama domain kustom `https://api.example.com` memiliki pemetaan API berikut.

Pemetaan API	API yang dipilih
(none)	API 1

Pemetaan API	API yang dipilih
orders	API 2
orders/v1/items	API 3
orders/v2/items	API 4
orders/v1/items/categories	API 5

Tabel berikut menunjukkan bagaimana API Gateway menerapkan pemetaan API sebelumnya ke permintaan contoh.

Permintaan	API yang dipilih	Penjelasan
https://api.example.com/orders	API 2	Permintaan sama persis dengan pemetaan API ini.
https://api.example.com/orders/v1/items	API 3	Permintaan sama persis dengan pemetaan API ini.
https://api.example.com/orders/v2/items	API 4	Permintaan sama persis dengan pemetaan API ini.
https://api.example.com/orders/v1/items/123	API 3	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang g. 123 Pada akhir permintaan tidak mempengaruhi pemilihan . Lihat the section called “Permintaan masuk ke nama domain kustom Anda” .
https://api.example.com/orders/v2/items/categories/5	API 5	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang.

Permintaan	API yang dipilih	Penjelasan
<code>https://api.example.com/customers</code>	API 1	API Gateway menggunakan pemetaan kosong sebagai tangkapan semua.
<code>https://api.example.com/ordersandmore</code>	API 2	API Gateway memilih pemetaan yang memiliki awalan pencocokan terpanjang. Untuk nama domain khusus yang dikonfigurasi dengan pemetaan tingkat tunggal, seperti hanya <code>https://api.example.com/orders</code> dan, API <code>https://api.example.com/Gateway</code> akan memilih API 1, karena tidak ada jalur yang cocok dengannya. <code>ordersandmore</code>

Pembatasan

- Dalam pemetaan API, nama domain khusus dan dipetakan APIs harus berada di akun yang sama AWS .
- Pemetaan API harus hanya berisi huruf, angka, dan karakter berikut: `$-_.+!*'()/`
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda dapat memiliki 200 pemetaan API dengan beberapa level untuk setiap nama domain. Batas ini tidak termasuk pemetaan API dengan level tunggal, seperti `/prod`.
- Anda hanya dapat memetakan HTTP APIs ke nama domain kustom regional dengan kebijakan keamanan TLS 1.2.
- Anda tidak dapat memetakan WebSocket APIs ke nama domain kustom yang sama dengan HTTP API atau REST API.

- Setelah membuat pemetaan API, Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS untuk dipetakan ke titik akhir API Anda.
- Jika Anda membuat pemetaan API dengan beberapa level, API Gateway mengonversi semua nama header menjadi huruf kecil.

Buat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Nama domain kustom Anda harus memiliki mode routing yang disetel ke salah satu `ROUTING_RULE_THEN_API_MAPPING` atau `API_MAPPING_ONLY`. Untuk informasi tentang cara mengatur mode perutean, lihat [the section called “Mengatur mode routing untuk nama domain kustom Anda”](#).

Misalnya AWS Serverless Application Model template yang membuat semua sumber daya, lihat [Sessions With SAM](#) on GitHub.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus dari panel navigasi utama.
3. Pilih nama domain khusus.
4. Pada tab Routing details, pilih Configure API mappings.
5. Masukkan API, Stage, dan Path untuk pemetaan.
6. Pilih Simpan.

AWS CLI

[create-api-mapping](#) Perintah berikut membuat pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan `api.example.com/v1/orders` ke API dan tahap yang ditentukan.

Note

Untuk membuat pemetaan API dengan beberapa level, Anda harus menggunakan `apigatewayv2`

```
aws apigatewayv2 create-api-mapping \
```

```
--domain-name api.example.com \  
--api-mapping-key v1/orders \  
--api-id a1b2c3d4 \  
--stage test
```

CloudFormation

CloudFormation Contoh berikut membuat pemetaan API.

Note

Untuk membuat pemetaan API dengan beberapa level, Anda harus menggunakan `AWS::ApiGatewayV2`.

```
MyApiMapping:  
  Type: 'AWS::ApiGatewayV2::ApiMapping'  
  Properties:  
    DomainName: api.example.com  
    ApiMappingKey: 'orders/v2/items'  
    ApiId: !Ref MyApi  
    Stage: !Ref MyStage
```

Jenis alamat IP untuk nama domain kustom di API Gateway

Saat Anda membuat nama domain khusus, Anda menentukan jenis alamat IP yang dapat memanggil domain Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk memanggil domain Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat memanggil domain Anda. Kami menyarankan Anda mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi titik akhir nama domain Anda.

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda.

- Jenis alamat IP default untuk nama domain kustom API Gateway untuk publik APIs adalah IPv4.
- Nama domain kustom pribadi hanya dapat memiliki tipe alamat IP dualstack.
- Nama domain kustom Anda tidak perlu memiliki jenis alamat IP yang sama untuk semua yang APIs dipetakan ke sana. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat memanggil domain Anda.

Mengubah jenis alamat IP dari nama domain kustom

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi titik akhir nama domain. Anda dapat memperbarui konfigurasi titik akhir dengan menggunakan Konsol Manajemen AWS,, AWS CLI CloudFormation, atau AWS SDK.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP dari nama domain kustom

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama domain kustom publik.
3. Pilih konfigurasi Endpoint.
4. Untuk jenis alamat IP, pilih salah satu IPv4 atau Dualstack.
5. Pilih Simpan.

AWS CLI

[update-domain-name](#) Perintah berikut memperbarui API untuk memiliki tipe alamat IP dualstack:

```
aws apigateway update-domain-name \
  --domain-name dualstack.example.com \
  --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

Outputnya akan terlihat seperti berikut:

```
{
  "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": {}
}
```

Pilih kebijakan keamanan untuk domain kustom Anda di API Gateway

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh API Gateway. Saat klien Anda membuat handshake TLS ke API atau nama domain kustom Anda, kebijakan keamanan memberlakukan versi TLS dan cipher suite yang diterima oleh API Gateway. Kebijakan keamanan melindungi nama domain Anda APIs dan kustom dari masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server.

API Gateway mendukung kebijakan keamanan lama dan kebijakan keamanan yang ditingkatkan. TLS_1_0 dan TLS_1_2 merupakan kebijakan keamanan warisan. Gunakan kebijakan keamanan ini untuk beban kerja umum, atau untuk mulai membuat API. Kebijakan apa pun yang dimulai SecurityPolicy_ adalah kebijakan keamanan yang ditingkatkan. Gunakan kebijakan ini untuk beban kerja yang diatur, tata kelola lanjutan, atau untuk menggunakan kriptografi pascakuantum. Jika Anda menggunakan kebijakan keamanan yang ditingkatkan, Anda juga harus menetapkan mode akses titik akhir untuk tata kelola tambahan. Untuk informasi selengkapnya, lihat [the section called “Mode akses titik akhir”](#).

Pertimbangan

Berikut ini adalah pertimbangan untuk kebijakan keamanan untuk nama domain kustom untuk REST APIs di API Gateway:

- Anda tidak dapat mengaktifkan TLS timbal balik pada nama domain yang menggunakan kebijakan keamanan yang disempurnakan.
- Anda tidak dapat memetakan API HTTP ke nama domain yang menggunakan kebijakan keamanan yang disempurnakan.

- Jika Anda mengaktifkan pemetaan jalur dasar multi-level ke REST API yang menggunakan kebijakan keamanan yang disempurnakan, Anda tidak dapat membuat pemetaan jalur dasar ke API HTTP untuk nama domain yang sama.
- API Anda dapat dipetakan ke nama domain khusus dengan kebijakan keamanan yang berbeda dari API Anda. Saat Anda memanggil nama domain kustom tersebut, API Gateway menggunakan kebijakan keamanan API untuk menegosiasikan jabat tangan TLS. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.
- API Gateway mendukung kebijakan keamanan pada semua APIs. Namun, Anda hanya dapat memilih kebijakan keamanan untuk REST APIs. API Gateway hanya mendukung kebijakan TLS_1_2 keamanan untuk HTTP atau WebSocket APIs.
- API Gateway tidak mendukung pembaruan kebijakan keamanan untuk nama domain dengan beberapa tipe titik akhir. Jika Anda memiliki beberapa jenis titik akhir untuk nama domain, hapus salah satunya untuk memperbarui kebijakan keamanan.

Bagaimana API Gateway menerapkan kebijakan keamanan

Contoh berikut menunjukkan bagaimana API Gateway menerapkan kebijakan keamanan menggunakan kebijakan `SecurityPolicy_TLS13_1_3_2025_09` keamanan sebagai contoh.

Kebijakan `SecurityPolicy_TLS13_1_3_2025_09` keamanan menerima lalu lintas TLS 1.3 dan menolak lalu lintas TLS 1.2 dan TLS 1.0. Untuk lalu lintas TLS 1.3, kebijakan keamanan menerima rangkaian sandi berikut:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway tidak menerima cipher suite lainnya. Misalnya, kebijakan keamanan akan menolak lalu lintas TLS 1.3 yang menggunakan AES128-SHA cipher suite.

Untuk memantau protokol TLS dan cipher klien yang digunakan untuk mengakses API Gateway Anda, Anda dapat menggunakan variabel `$context.tlsVersion` dan `$context.cipherSuite` konteks dalam log akses Anda. Untuk informasi selengkapnya, lihat [the section called “Memantau”](#).

Untuk melihat kebijakan keamanan default untuk semua REST APIs dan nama domain kustom, lihat [the section called “Kebijakan keamanan default”](#). Untuk melihat kebijakan keamanan yang

didukung untuk semua REST APIs dan nama domain kustom, lihat [the section called “Kebijakan keamanan yang didukung”](#).

Ubah kebijakan keamanan nama domain kustom Anda

Jika Anda mengubah kebijakan keamanan Anda, dibutuhkan sekitar 15 menit untuk pembaruan selesai. Anda dapat memantau `lastUpdateStatus` nama domain kustom Anda. Saat nama domain kustom Anda diperbarui, `lastUpdateStatus` adalah `PENDING` dan ketika selesai, itu akan menjadi `AVAILABLE`.

Saat Anda menggunakan kebijakan keamanan yang dimulai `SecurityPolicy_`, Anda juga harus mengaktifkan mode akses titik akhir. Untuk informasi selengkapnya, lihat [the section called “Mode akses titik akhir”](#).

Konsol Manajemen AWS

Untuk mengubah kebijakan keamanan nama domain kustom

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama domain khusus yang mengirimkan lalu lintas ke REST APIs.

Pastikan hanya ada satu jenis endpoint yang terkait dengan nama domain kustom Anda.

3. Pilih Pengaturan nama domain khusus, lalu pilih Edit.
4. Untuk kebijakan Keamanan, pilih kebijakan baru.
5. Untuk mode akses Endpoint, pilih Strict.
6. Pilih Simpan perubahan.

AWS CLI

[update-domain-name](#) Perintah berikut memperbarui nama domain untuk menggunakan kebijakan `SecurityPolicy_TLS13_1_3_2025_09` keamanan:

```
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"
      }
    ]'

```

Outputnya akan terlihat seperti berikut:

```

{
  "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"
}

```

Informasi tentang HTTP APIs dan WebSocket APIs

Untuk informasi lebih lanjut tentang HTTP APIs dan WebSocket APIs, lihat [the section called “Kebijakan keamanan untuk HTTP APIs di API Gateway”](#) dan [the section called “Kebijakan keamanan untuk WebSocket APIs di API Gateway”](#).

Nonaktifkan titik akhir default untuk REST APIs

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus, nonaktifkan `execute-api` titik akhir default. Klien masih dapat terhubung ke titik akhir default Anda, tetapi mereka akan menerima kode 403 Forbidden status. Menonaktifkan titik akhir default memengaruhi semua tahapan API. Pengaturan ini akan berpengaruh saat Anda memperbarui pengaturan apa pun pada tahap apa pun, seperti memperbarui penerapan di panggung.

Prosedur berikut menunjukkan cara menonaktifkan titik akhir default untuk REST API.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Pada panel navigasi utama, pilih pengaturan API.
4. Pilih API.
5. Pada detail API, pilih Edit.
6. Untuk titik akhir Default, pilih Tidak Aktif.
7. Pilih Simpan perubahan.
8. Pada panel navigasi utama, pilih Resources.
9. Pilih Deploy API.
10. Menerapkan ulang API Anda ke panggung atau perbarui setelan apa pun di panggung agar pembaruan diterapkan.

AWS CLI

[update-rest-api](#) Perintah berikut menonaktifkan titik akhir default:

```
aws apigateway update-rest-api \  
  --rest-api-id abcdef123 \  
  --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Setelah menonaktifkan titik akhir default, Anda harus menerapkan API agar perubahan diterapkan.

Perintah [create-deployment](#) berikut membuat deployment dan mengaitkannya dengan tahapan:

```
aws apigateway create-deployment \  
  --rest-api-id abcdef123 \  
  --stage-name dev
```

Konfigurasi pemeriksaan kesehatan khusus untuk failover DNS untuk API Gateway API

Anda dapat menggunakan pemeriksaan kesehatan Amazon Route 53 untuk mengontrol failover DNS dari API Gateway API di primer Wilayah AWS ke salah satu di Wilayah sekunder. Ini dapat

membantu mengurangi dampak jika terjadi masalah Regional. Jika Anda menggunakan domain kustom, Anda dapat melakukan failover tanpa mengharuskan klien untuk mengubah titik akhir API.

Bila Anda memilih [Evaluasi Kesehatan Target](#) untuk catatan alias, catatan tersebut gagal hanya jika layanan API Gateway tidak tersedia di Wilayah. Dalam beberapa kasus, API Gateway Anda sendiri APIs dapat mengalami gangguan sebelum waktu itu. Untuk mengontrol failover DNS secara langsung, konfigurasi pemeriksaan kesehatan Route 53 kustom untuk API Gateway Anda. APIs Untuk contoh ini, Anda menggunakan CloudWatch alarm yang membantu operator mengontrol failover DNS. Untuk lebih banyak contoh dan pertimbangan lain saat Anda mengonfigurasi failover, lihat [Membuat Mekanisme Pemulihan Bencana Menggunakan Rute 53](#) dan [Melakukan pemeriksaan kesehatan Route 53 pada sumber daya pribadi di VPC](#) dengan dan. AWS Lambda CloudWatch

Topik

- [Prasyarat](#)
- [Langkah 1: Siapkan sumber daya](#)
- [Langkah 2: Memulai failover ke Wilayah sekunder](#)
- [Langkah 3: Uji failover](#)
- [Langkah 4: Kembali ke wilayah utama](#)
- [Langkah selanjutnya: Sesuaikan dan uji secara teratur](#)

Prasyarat

Untuk menyelesaikan prosedur ini, Anda harus membuat dan mengonfigurasi sumber daya berikut:

- Nama domain yang Anda miliki.
- Sertifikat ACM untuk nama domain itu menjadi dua Wilayah AWS. Untuk info lebih lanjut, lihat [the section called “Prasyarat untuk nama domain khusus”](#).
- Zona yang dihosting Route 53 untuk nama domain Anda. Untuk informasi selengkapnya, lihat [Bekerja dengan zona yang dihosting](#) di Panduan Pengembang Amazon Route 53.

Untuk informasi selengkapnya tentang cara membuat catatan DNS failover Route 53 untuk nama domain, lihat [Memilih kebijakan perutean](#) di Panduan Pengembang Amazon Route 53. Untuk informasi selengkapnya tentang cara memantau CloudWatch alarm, lihat [Memantau CloudWatch alarm](#) di Panduan Pengembang Amazon Route 53.

Langkah 1: Siapkan sumber daya

Dalam contoh ini, Anda membuat sumber daya berikut untuk mengonfigurasi failover DNS untuk nama domain Anda:

- API Gateway APIs menjadi dua Wilayah AWS
- API Gateway nama domain kustom dengan nama yang sama dalam dua Wilayah AWS
- Pemetaan API Gateway API yang menghubungkan API Gateway Anda APIs ke nama domain khusus
- Route 53 catatan DNS failover untuk nama domain
- CloudWatch Alarm di Wilayah sekunder
- Pemeriksaan kesehatan Route 53 berdasarkan CloudWatch alarm di Wilayah sekunder

Pertama, pastikan Anda memiliki semua sumber daya yang diperlukan di Wilayah primer dan sekunder. Wilayah sekunder harus berisi alarm dan pemeriksaan kesehatan. Dengan cara ini, Anda tidak bergantung pada Wilayah utama untuk melakukan failover. Misalnya CloudFormation template yang membuat sumber daya ini, lihat [primary.yaml](#) dan [secondary.yaml](#).

Important

Sebelum failover ke Wilayah sekunder, pastikan bahwa semua sumber daya yang diperlukan tersedia. Jika tidak, API Anda tidak akan siap untuk lalu lintas di Wilayah sekunder.

Langkah 2: Memulai failover ke Wilayah sekunder

Dalam contoh berikut, Region siaga menerima CloudWatch metrik dan memulai failover. Kami menggunakan metrik khusus yang memerlukan intervensi operator untuk memulai failover.

```
aws cloudwatch put-metric-data \  
  --metric-name Failover \  
  --namespace HealthCheck \  
  --unit Count \  
  --value 1 \  
  --region us-west-1
```

Ganti data metrik dengan data yang sesuai untuk CloudWatch alarm yang Anda konfigurasi.

Langkah 3: Uji failover

Panggil API Anda dan verifikasi bahwa Anda mendapatkan respons dari Wilayah sekunder. Jika Anda menggunakan contoh templat di langkah 1, respons berubah dari {"message": "Hello from the primary Region!"} ke {"message": "Hello from the secondary Region!"} setelah failover.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

Langkah 4: Kembali ke wilayah utama

Untuk kembali ke Wilayah utama, kirim CloudWatch metrik yang menyebabkan pemeriksaan kesehatan lulus.

```
aws cloudwatch put-metric-data \
  --metric-name Failover \
  --namespace HealthCheck \
  --unit Count \
  --value 0 \
  --region us-west-1
```

Ganti data metrik dengan data yang sesuai untuk CloudWatch alarm yang Anda konfigurasi.

Panggil API Anda dan verifikasi bahwa Anda mendapatkan respons dari Wilayah utama. Jika Anda menggunakan contoh templat di langkah 1, respons berubah dari {"message": "Hello from the secondary Region!"} ke {"message": "Hello from the primary Region!"}.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Langkah selanjutnya: Sesuaikan dan uji secara teratur

Contoh ini menunjukkan salah satu cara untuk mengkonfigurasi failover DNS. Anda dapat menggunakan berbagai CloudWatch metrik atau titik akhir HTTP untuk pemeriksaan kesehatan yang mengelola failover. Uji mekanisme failover Anda secara teratur untuk memastikan bahwa mereka bekerja seperti yang diharapkan, dan operator terbiasa dengan prosedur failover Anda.

Optimalkan kinerja REST APIs

Setelah Anda membuat API tersedia untuk dipanggil, Anda mungkin menyadari bahwa itu perlu dioptimalkan untuk meningkatkan daya tanggap. API Gateway menyediakan beberapa strategi untuk mengoptimalkan API Anda, seperti cache respons dan kompresi payload. Di bagian ini, Anda dapat mempelajari cara mengaktifkan kemampuan ini.

Topik

- [Pengaturan cache untuk REST APIs di API Gateway](#)
- [Kompresi muatan untuk REST APIs di API Gateway](#)

Pengaturan cache untuk REST APIs di API Gateway

Anda dapat mengaktifkan caching API di API Gateway untuk men-cache respons titik akhir Anda. Dengan caching, Anda dapat mengurangi jumlah panggilan yang dilakukan ke titik akhir Anda dan juga meningkatkan latensi permintaan ke API Anda.

Saat Anda mengaktifkan caching untuk suatu tahap, API Gateway menyimpan respons dari titik akhir Anda untuk periode tertentu time-to-live (TTL), dalam hitungan detik. API Gateway kemudian merespons permintaan dengan mencari respons titik akhir dari cache alih-alih membuat permintaan ke titik akhir Anda. Nilai TTL default untuk caching API adalah 300 detik. Nilai TTL maksimum adalah 3600 detik. TTL = 0 berarti caching dinonaktifkan.

Note

Caching adalah upaya terbaik. Anda dapat menggunakan CacheMissCount metrik CacheHitCount dan di Amazon CloudWatch untuk memantau permintaan yang disajikan API Gateway dari cache API.

Ukuran maksimum respons yang dapat di-cache adalah 1048576 byte. Enkripsi data cache dapat meningkatkan ukuran respons saat sedang di-cache.

Ini adalah Layanan yang Memenuhi Syarat HIPAA. [Untuk informasi lebih lanjut tentang AWS, Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS tahun 1996 \(HIPAA\), dan menggunakan AWS layanan untuk memproses, menyimpan, dan mengirimkan informasi kesehatan yang dilindungi \(PHI\), lihat Ikhtisar HIPAA.](#)

 Important

Saat Anda mengaktifkan caching untuk suatu tahap, hanya GET metode yang mengaktifkan caching secara default. Ini membantu memastikan keamanan dan ketersediaan API Anda. Anda dapat mengaktifkan caching untuk metode lain dengan [mengganti pengaturan metode](#).

 Important

Caching dibebankan per jam berdasarkan ukuran cache yang Anda pilih. Caching tidak memenuhi syarat untuk Tingkat AWS Gratis. Untuk informasi selengkapnya, lihat [Harga API Gateway](#).

Aktifkan caching Amazon API Gateway

Di API Gateway, Anda dapat mengaktifkan caching untuk tahap tertentu.

Ketika Anda mengaktifkan caching, Anda harus memilih kapasitas cache. Secara umum, kapasitas yang lebih besar memberikan kinerja yang lebih baik, tetapi juga lebih mahal. Untuk ukuran cache yang didukung, lihat [cacheClusterSize](#) di Referensi API Gateway API.

API Gateway memungkinkan caching dengan membuat instance cache khusus. Proses ini bisa memakan waktu hingga 4 menit.

API Gateway mengubah kapasitas caching dengan menghapus instance cache yang ada dan membuat yang baru dengan kapasitas yang dimodifikasi. Semua data cache yang ada dihapus.

 Note

Kapasitas cache mempengaruhi CPU, memori, dan bandwidth jaringan dari instance cache. Akibatnya, kapasitas cache dapat memengaruhi kinerja cache Anda.

API Gateway merekomendasikan agar Anda menjalankan uji pemuatan 10 menit untuk memverifikasi bahwa kapasitas cache sesuai dengan beban kerja Anda. Pastikan bahwa lalu lintas selama uji beban mencerminkan lalu lintas produksi. Misalnya, sertakan ramp up, lalu lintas konstan, dan lonjakan lalu lintas. Tes beban harus mencakup respons yang dapat disajikan dari cache, serta respons unik yang menambahkan item ke cache. Pantau metrik latensi, 4xx, 5xx, cache hit, dan cache miss selama uji beban. Sesuaikan kapasitas

cache sesuai kebutuhan berdasarkan metrik ini. Untuk informasi selengkapnya tentang pengujian beban, lihat [Bagaimana cara memilih kapasitas cache API Gateway terbaik agar tidak mencapai batas laju?](#) .

Konsol Manajemen AWS

Di konsol API Gateway, Anda mengonfigurasi caching di halaman Tahapan. Anda menyediakan cache tahap dan menentukan pengaturan cache tingkat metode default. Jika Anda mengaktifkan cache tingkat metode default, caching tingkat metode diaktifkan untuk semua GET metode di panggung Anda, kecuali metode tersebut memiliki penggantian metode. GETMetode tambahan apa pun yang Anda terapkan ke tahap Anda akan memiliki cache tingkat metode. Untuk mengonfigurasi pengaturan caching tingkat metode untuk metode spesifik tahap Anda, Anda dapat menggunakan penggantian metode. Untuk informasi selengkapnya tentang penggantian metode, lihat. [the section called “Ganti caching tahap untuk caching metode”](#)

Untuk mengonfigurasi caching API untuk tahap tertentu:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Memilih Tahapan.
3. Dalam daftar Tahapan untuk API, pilih stage.
4. Di bagian Detail tahap, pilih Edit.
5. Di bawah Pengaturan tambahan, untuk pengaturan Cache, aktifkan cache API Penyediaan.

Ini menyediakan cluster cache untuk tahap Anda.

6. Untuk mengaktifkan caching untuk tahap Anda, aktifkan caching tingkat metode Default.

Ini mengaktifkan caching tingkat metode untuk semua GET metode di panggung Anda. GETMetode tambahan apa pun yang Anda terapkan ke tahap ini akan memiliki cache tingkat metode.

Note

Jika Anda memiliki setelan yang ada untuk cache tingkat metode, mengubah pengaturan caching tingkat metode default tidak memengaruhi pengaturan yang ada.

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. Pilih Simpan perubahan.

AWS CLI

Perintah [tahap pembaruan berikut memperbarui tahapan](#) untuk menyediakan cache dan mengaktifkan caching tingkat metode untuk semua metode di panggung Anda: GET

```
aws apigateway update-stage \
  --rest-api-id a1b2c3 \
  --stage-name 'prod' \
  --patch-operations file://patch.json
```

Isi dari `patch.json` adalah sebagai berikut:

```
[
  {
    "op": "replace",
    "path": "/cacheClusterEnabled",
    "value": "true"
  },
  {
    "op": "replace",
    "path": "/cacheClusterSize",
    "value": "0.5"
  },
  {
    "op": "replace",
    "path": "/*/*/caching/enabled",
    "value": "true"
  }
]
```

]

Note

Jika Anda memiliki setelan yang ada untuk cache tingkat metode, mengubah pengaturan caching tingkat metode default tidak memengaruhi pengaturan yang ada.

Note

Membuat atau menghapus cache membutuhkan waktu sekitar 4 menit agar API Gateway selesai.

Ketika cache dibuat, nilai cluster Cache berubah dari `Create in progress` ke `Active`. Ketika penghapusan cache selesai, nilai cluster Cache berubah dari `Delete in progress` ke `Inactive`.

Saat Anda mengaktifkan caching tingkat metode untuk semua metode di panggung Anda, nilai caching tingkat metode Default berubah menjadi `Active`. Jika Anda menonaktifkan caching tingkat metode untuk semua metode di panggung Anda, nilai caching tingkat metode Default akan berubah menjadi `Inactive`. Jika Anda memiliki setelan yang ada untuk cache tingkat metode, mengubah status cache tidak memengaruhi pengaturan tersebut.

Saat Anda mengaktifkan caching dalam pengaturan Cache tahap, hanya GET metode yang di-cache. Untuk memastikan keamanan dan ketersediaan API Anda, sebaiknya jangan mengubah setelan ini. Namun, Anda dapat mengaktifkan caching untuk metode lain dengan [mengganti pengaturan metode](#).

Jika Anda ingin memverifikasi apakah caching berfungsi seperti yang diharapkan, Anda memiliki dua opsi umum:

- Periksa CloudWatch metrik `CacheHitCount` dan `CacheMissCount` untuk API dan panggung Anda.
- Masukkan stempel waktu dalam respons.

Note

Jangan gunakan `X-Cache` header dari CloudFront respons untuk menentukan apakah API Anda sedang dilayani dari instance cache API Gateway Anda.

Ganti caching tingkat tahap API Gateway untuk caching tingkat metode

Anda dapat mengganti pengaturan cache tingkat tahap dengan mengaktifkan atau mematikan caching untuk metode tertentu. Anda juga dapat memodifikasi periode TTL atau mengaktifkan atau menonaktifkan enkripsi untuk respons yang di-cache.

Jika Anda mengantisipasi bahwa metode yang Anda caching akan menerima data sensitif dalam tanggapannya, enkripsi data cache Anda. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Konsol Manajemen AWS

Jika Anda mengubah pengaturan caching tingkat metode default di detail Stage, itu tidak memengaruhi pengaturan cache tingkat metode yang memiliki penggantian.

Jika Anda mengantisipasi bahwa metode yang Anda caching akan menerima data sensitif dalam tanggapannya, di Pengaturan Cache, pilih Enkripsi data cache.

Untuk mengonfigurasi caching API untuk metode individual menggunakan konsol:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API.
3. Memilih Tahapan.
4. Dalam daftar Tahapan untuk API, perluas tahap dan pilih metode di API.
5. Di bagian Penggantian Metode, pilih Edit.
6. Di bagian Pengaturan metode, aktifkan atau matikan Aktifkan cache metode atau sesuaikan opsi lain yang diinginkan.



Note

Caching tidak aktif sampai Anda menyediakan cluster cache untuk tahap Anda.

7. Pilih Simpan.

AWS CLI

Perintah [update-stage](#) berikut mematikan cache hanya untuk metode ini: GET /pets

```
aws apigateway update-stage /  
  --rest-api-id a1b2c3 /  
  --stage-name 'prod' /  
  --patch-operations file://patch.json
```

Isi dari `patch.json` adalah sebagai berikut:

```
[{  
  "op": "replace",  
  "path": "/~1pets/GET/caching/enabled",  
  "value": "false"  
}]
```

Gunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache

Anda dapat menggunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache. Ini termasuk header khusus, jalur URL, atau string kueri. Anda dapat menentukan beberapa atau semua parameter ini sebagai kunci cache, tetapi Anda harus menentukan setidaknya satu nilai. Bila Anda memiliki kunci cache, API Gateway menyimpan respons dari setiap nilai kunci secara terpisah, termasuk saat kunci cache tidak ada.

 Note

Kunci cache diperlukan saat mengatur caching pada sumber daya.

Misalnya, Anda memiliki permintaan dalam format berikut:

```
GET /users?type=... HTTP/1.1  
host: example.com  
...
```

Dalam permintaan ini, `type` dapat mengambil nilai `admin` atau `regular`. Jika Anda menyertakan `type` parameter sebagai bagian dari kunci cache, respons dari di-cache `GET /users?type=admin` secara terpisah dari yang dari `GET /users?type=regular`.

Ketika sebuah metode atau permintaan integrasi mengambil lebih dari satu parameter, Anda dapat memilih untuk menyertakan beberapa atau semua parameter untuk membuat kunci cache. Misalnya, Anda hanya dapat menyertakan `type` parameter dalam kunci cache untuk permintaan berikut, dibuat dalam urutan yang tercantum dalam periode TTL:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

Respons dari permintaan ini di-cache dan digunakan untuk melayani permintaan berikut:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Konsol Manajemen AWS

Untuk menyertakan metode atau parameter permintaan integrasi sebagai bagian dari kunci cache di konsol API Gateway, pilih **Caching** setelah Anda menambahkan parameter.

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

Perintah [put-method](#) berikut menciptakan GET metode dan membutuhkan type parameter string query:

```
aws apigateway put-method /
  --rest-api-id a1b2c3 /
  --resource-id aaa111 /
  --http-method GET /
  --authorization-type "NONE" /
  --request-parameters "method.request.querystring.type=true"
```

Perintah [put-integration](#) berikut membuat integrasi untuk GET metode dengan titik akhir HTTP dan menetapkan bahwa API Gateway menyimpan parameter permintaan metode: type

```
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"
```

Untuk menentukan cache API Gateway parameter permintaan integrasi, gunakan `integration.request.location.name` sebagai parameter kunci cache.

Siram cache tahap API di API Gateway

Saat caching API diaktifkan, Anda dapat membersihkan cache tahap API Anda untuk memastikan bahwa klien API Anda mendapatkan respons terbaru dari titik akhir integrasi Anda.

Konsol Manajemen AWS

Untuk menyiram cache tahap API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API yang memiliki tahapan dengan cache.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih panggung Anda dengan cache.
4. Pilih menu Stage actions, dan kemudian pilih Flush stage cache.

AWS CLI

[flush-stage-cache](#) Perintah berikut membersihkan cache panggung:

```
aws apigateway flush-stage-cache \  
  --rest-api-id a1b2c3 \  
  --stage-name prod
```

Note

Setelah cache dimatikan, respons dilayani dari titik akhir integrasi hingga cache dibangun kembali. Selama periode ini, jumlah permintaan yang dikirim ke titik akhir integrasi dapat meningkat. Ini dapat meningkatkan latensi keseluruhan API untuk sementara waktu.

Membatalkan entri cache API Gateway

Klien API Anda dapat membatalkan entri cache yang ada dan memuatnya kembali dari titik akhir integrasi untuk permintaan individual. Klien harus mengirim permintaan yang berisi Cache-Control: max-age=0 header. Klien menerima respons langsung dari titik akhir integrasi alih-alih cache, asalkan klien berwenang untuk melakukannya. Ini menggantikan entri cache yang ada dengan respons baru, yang diambil dari titik akhir integrasi.

Untuk memberikan izin kepada klien, lampirkan kebijakan format berikut ke peran eksekusi IAM bagi pengguna.

Note

Pembatalan cache lintas akun tidak didukung.

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"
      ]
    }
  ]
}
```

Kebijakan ini memungkinkan layanan eksekusi API Gateway membatalkan cache untuk permintaan pada sumber daya tertentu (atau sumber daya). Untuk menentukan sekelompok sumber daya yang ditargetkan, gunakan karakter wildcard (*) untuk *account-id* *api-id*, dan entri lain dalam nilai ARN. Resource Untuk informasi selengkapnya tentang cara menyetel izin untuk layanan eksekusi API Gateway, lihat [Kontrol akses ke REST API dengan izin IAM](#).

Jika Anda tidak memaksakan `InvalidateCache` kebijakan (atau memilih kotak centang `Memerlukan otorisasi` di konsol), klien mana pun dapat membatalkan cache API. Jika sebagian besar atau semua klien membatalkan cache API, ini dapat meningkatkan latensi API Anda secara signifikan.

Saat kebijakan diberlakukan, caching diaktifkan dan otorisasi diperlukan.

Anda dapat menentukan cara API Gateway menangani permintaan yang tidak sah dengan memilih dari opsi berikut:

Gagal permintaan dengan kode status 403

API Gateway mengembalikan `403 Unauthorized` respons.

Untuk mengatur opsi ini menggunakan API, gunakan `FAIL_WITH_403`.

Abaikan header kontrol cache; Tambahkan peringatan di header respons

API Gateway memproses permintaan dan menambahkan header peringatan dalam respons.

Untuk mengatur opsi ini menggunakan API, gunakan `SUCCEED_WITH_RESPONSE_HEADER`.

Abaikan header kontrol cache

API Gateway memproses permintaan dan tidak menambahkan header peringatan dalam respons.

Untuk mengatur opsi ini menggunakan API, gunakan `SUCCEED_WITHOUT_RESPONSE_HEADER`.

Anda dapat menyetel perilaku penanganan permintaan yang tidak sah menggunakan konsol API Gateway atau AWS CLI.

Konsol Manajemen AWS

Untuk menentukan bagaimana permintaan yang tidak sah ditangani

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API yang memiliki tahapan dengan cache.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih panggung Anda dengan cache.
4. Untuk detail Stage, pilih Edit.
5. Untuk penanganan permintaan yang tidak sah, pilih opsi.
6. Pilih Lanjutkan.

7. Tinjau perubahan Anda dan pilih Simpan perubahan.

AWS CLI

Perintah [update-stage](#) berikut memperbarui tahapan untuk menangani permintaan yang tidak sah dengan gagal permintaan dengan kode status 403:

```
aws apigateway update-stage /  
  --rest-api-id a1b2c3 /  
  --stage-name 'prod' /  
  --patch-operations 'op=replace,path=/**/caching/  
unauthorizedCacheControlHeaderStrategy,value="FAIL_WITH_403"'
```

CloudFormation contoh tahap dengan cache

CloudFormation Template berikut membuat contoh API, menyediakan cache 0.5 GB untuk Prod panggung, dan mengaktifkan caching tingkat metode untuk semua metode. GET

Important

Caching dibebankan per jam berdasarkan ukuran cache yang Anda pilih. Caching tidak memenuhi syarat untuk Tingkat AWS Gratis. Untuk informasi selengkapnya, lihat [Harga API Gateway](#).

Contoh CloudFormation template

```
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

```

Kompresi muatan untuk REST APIs di API Gateway

API Gateway memungkinkan klien Anda memanggil API Anda dengan muatan terkompresi dengan menggunakan salah satu pengkodean [konten yang didukung](#). Secara default, API Gateway mendukung dekompresi payload permintaan metode. Namun, Anda harus mengonfigurasi API Anda untuk mengaktifkan kompresi payload respons metode.

Untuk mengaktifkan kompresi pada [API](#), setel [minimumCompressionsSize](#) properti ke bilangan bulat non-negatif antara 0 dan 10485760 (10M byte) saat Anda membuat API atau setelah Anda membuat API. Untuk menonaktifkan kompresi pada API, atur `minimumCompressionSize` ke null

atau hapus sama sekali. Anda dapat mengaktifkan atau menonaktifkan kompresi untuk API dengan menggunakan konsol API Gateway, API AWS CLI, atau API Gateway REST API.

Jika Anda ingin kompresi diterapkan pada muatan dengan ukuran berapa pun, atur `minimumCompressionSize` nilainya ke nol. Namun, mengompresi data dengan ukuran kecil sebenarnya dapat meningkatkan ukuran data akhir. Selain itu, kompresi di API Gateway dan dekompresi di klien dapat meningkatkan latensi keseluruhan dan membutuhkan lebih banyak waktu komputasi. Anda harus menjalankan kasus pengujian terhadap API Anda untuk menentukan nilai optimal.

Klien dapat mengirimkan permintaan API dengan muatan terkompresi dan `Content-Encoding` header yang sesuai untuk API Gateway untuk mendekompresi dan menerapkan templat pemetaan yang berlaku, sebelum meneruskan permintaan ke titik akhir integrasi. Setelah kompresi diaktifkan dan API diterapkan, klien dapat menerima respons API dengan muatan terkompresi jika menentukan `Accept-Encoding` header yang sesuai dalam permintaan metode.

Saat titik akhir integrasi mengharapkan dan mengembalikan muatan JSON yang tidak terkompresi, templat pemetaan apa pun yang dikonfigurasi untuk muatan JSON yang tidak terkompresi berlaku untuk muatan terkompresi. Untuk payload permintaan metode terkompresi, API Gateway mendekompresi payload, menerapkan template pemetaan, dan meneruskan permintaan yang dipetakan ke titik akhir integrasi. Untuk payload respons integrasi yang tidak terkompresi, API Gateway menerapkan template pemetaan, memampatkan payload yang dipetakan, dan mengembalikan payload terkompresi ke klien.

Topik

- [Aktifkan kompresi payload untuk API di API Gateway](#)
- [Panggil metode API dengan payload terkompresi di API Gateway](#)
- [Menerima respons API dengan payload terkompresi di API Gateway](#)

Aktifkan kompresi payload untuk API di API Gateway

Anda dapat mengaktifkan kompresi untuk API menggunakan konsol API Gateway, the AWS CLI, atau AWS SDK.

Untuk API yang ada, Anda harus menerapkan API setelah mengaktifkan kompresi agar perubahan diterapkan. Untuk API baru, Anda dapat menerapkan API setelah penyiapan API selesai.

 Note

Pengkodean konten dengan prioritas tertinggi harus didukung oleh API Gateway. Jika tidak, kompresi tidak diterapkan pada muatan respons.

Topik

- [Mengaktifkan kompresi payload untuk API menggunakan konsol API Gateway](#)
- [Aktifkan kompresi payload untuk API menggunakan AWS CLI](#)
- [Pengkodean konten yang didukung oleh API Gateway](#)

Mengaktifkan kompresi payload untuk API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara mengaktifkan kompresi payload untuk API.

Untuk mengaktifkan kompresi payload dengan menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih pengaturan API.
4. Di bagian detail API, pilih Edit.
5. Aktifkan pengkodean konten untuk mengaktifkan kompresi muatan. Untuk ukuran tubuh Minimum, masukkan angka untuk ukuran kompresi minimum (dalam byte). Untuk mematikan kompresi, matikan opsi Pengkodean konten.
6. Pilih Simpan perubahan.

Aktifkan kompresi payload untuk API menggunakan AWS CLI

[create-rest-api](#) Perintah berikut membuat API dengan kompresi payload:

```
aws apigateway create-rest-api \  
  --name "My test API" \  
  --minimum-compression-size 0
```

[update-rest-api](#) Perintah berikut memungkinkan kompresi payload untuk API yang ada:

```
aws apigateway update-rest-api \  
  --rest-api-id 1234567890 \  
  --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

`minimumCompressionSize` properti memiliki nilai integer non-negatif antara 0 dan 10485760 (10M byte). Ini mengukur ambang kompresi. Jika ukuran muatan lebih kecil dari nilai ini, kompresi atau dekompresi tidak diterapkan pada muatan. Pengaturan ke nol memungkinkan kompresi untuk ukuran muatan apa pun.

[update-rest-api](#) Perintah berikut mematikan kompresi payload:

```
aws apigateway update-rest-api \  
  --rest-api-id 1234567890 \  
  --patch-operations op=replace,path=/minimumCompressionSize,value=
```

Anda juga dapat mengatur `value` ke string kosong `""` atau menghilangkan `value` properti sama sekali dalam panggilan sebelumnya.

Pengkodean konten yang didukung oleh API Gateway

API Gateway mendukung pengkodean konten berikut:

- `deflate`
- `gzip`
- `identity`

API Gateway juga mendukung format `Accept-Encoding` header berikut, sesuai dengan spesifikasi [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`

Panggil metode API dengan payload terkompresi di API Gateway

Untuk membuat permintaan API dengan muatan terkompresi, klien harus menyetel Content-Encoding header dengan salah satu pengkodean [konten yang didukung](#).

Misalkan Anda adalah klien API dan ingin memanggil metode PetStore API (POST /pets). Jangan panggil metode dengan menggunakan output JSON berikut:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

Sebagai gantinya, Anda dapat memanggil metode dengan muatan yang sama dikompresi dengan menggunakan pengkodean GZIP:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

   RPP* ,HU RPJ 0W  e&   L, ,-y j
```

Ketika API Gateway menerima permintaan, itu memverifikasi apakah pengkodean konten yang ditentukan didukung. Kemudian, ia mencoba untuk mendekompresi muatan dengan pengkodean konten yang ditentukan. Jika dekompresi berhasil, ia mengirimkan permintaan ke titik akhir integrasi. Jika pengkodean yang ditentukan tidak didukung atau muatan yang disediakan tidak dikompresi dengan pengkodean tertentu, API Gateway mengembalikan respons 415 Unsupported Media Type kesalahan. Kesalahan tidak dicatat ke CloudWatch Log, jika terjadi pada fase awal dekompresi sebelum API dan tahap Anda diidentifikasi.

Menerima respons API dengan payload terkompresi di API Gateway

[Saat membuat permintaan pada API yang mendukung kompresi, klien dapat memilih untuk menerima muatan respons terkompresi dari format tertentu dengan menentukan Accept-Encoding header dengan pengkodean konten yang didukung.](#)

API Gateway hanya memampatkan payload respons jika kondisi berikut terpenuhi:

- Permintaan masuk memiliki Accept-Encoding header dengan pengkodean dan format konten yang didukung.

 Note

Jika header tidak disetel, nilai default adalah * seperti yang didefinisikan dalam [RFC 7231](#). Dalam kasus seperti itu, API Gateway tidak memampatkan muatan. Beberapa browser atau klien dapat menambahkan Accept-Encoding (misalnya, Accept-Encoding: gzip, deflate, br) secara otomatis ke permintaan yang diaktifkan kompresi. Ini dapat mengaktifkan kompresi payload di API Gateway. Tanpa spesifikasi eksplisit dari nilai Accept-Encoding header yang didukung, API Gateway tidak memampatkan payload.

- `minimumCompressionSize` ini diatur pada API untuk mengaktifkan kompresi.
- Respons integrasi tidak memiliki Content-Encoding header.
- Ukuran muatan respons integrasi, setelah templat pemetaan yang berlaku diterapkan, lebih besar dari atau sama dengan nilai yang ditentukan `minimumCompressionSize`.

API Gateway menerapkan template pemetaan apa pun yang dikonfigurasi untuk respons integrasi sebelum mengompresi payload. Jika respons integrasi berisi Content-Encoding header, API Gateway mengasumsikan bahwa payload respons integrasi sudah dikompresi dan melewati pemrosesan kompresi.

Contohnya adalah contoh PetStore API dan permintaan berikut:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

Backend merespons permintaan dengan muatan JSON yang tidak terkompresi yang mirip dengan yang berikut ini:

```
200 OK
```

```
[
  {
```

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

Untuk menerima output ini sebagai payload terkompresi, klien API Anda dapat mengirimkan permintaan sebagai berikut:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Klien menerima respons dengan Content-Encoding header dan muatan yang dikodekan GZIP yang mirip dengan berikut ini:

```
200 OK
Content-Encoding:gzip
...
R
J)V
:P^IeA*+++++(L XXYZku0L0B7!9CC#&YYYa^^X
```

Ketika payload respons dikompresi, hanya ukuran data terkompresi yang ditagih untuk transfer data.

Mendistribusikan REST Anda APIs ke klien di API Gateway

Bagian ini memberikan detail tentang mendistribusikan API Gateway Anda APIs kepada pelanggan Anda. Mendistribusikan API Anda termasuk membuat SDKs pelanggan Anda mengunduh dan

mengintegrasikan dengan aplikasi klien mereka, mendokumentasikan API Anda sehingga pelanggan tahu cara memanggilnya dari aplikasi klien mereka, dan membuat API Anda tersedia sebagai bagian dari penawaran produk.

Topik

- [Paket penggunaan dan kunci API untuk REST APIs di API Gateway](#)
- [Dokumentasi untuk REST APIs di API Gateway](#)
- [Hasilkan SDKs untuk REST APIs di API Gateway](#)
- [Jual API Gateway Anda APIs melalui AWS Marketplace](#)

Paket penggunaan dan kunci API untuk REST APIs di API Gateway

Setelah membuat, menguji, dan menerapkan APIs, Anda dapat menggunakan paket penggunaan API Gateway untuk membuatnya tersedia sebagai penawaran produk bagi pelanggan Anda. Anda dapat mengonfigurasi paket penggunaan dan kunci API untuk memungkinkan pelanggan mengakses yang dipilih APIs, dan mulai membatasi permintaan APIs berdasarkan batas dan kuota yang ditentukan. Ini dapat diatur pada API, atau tingkat metode API.

Apa itu paket penggunaan dan kunci API?

Paket penggunaan menentukan siapa yang dapat mengakses satu atau beberapa tahapan dan metode API yang disebar—dan secara opsional menetapkan tingkat permintaan target untuk memulai pembatasan permintaan. Paket menggunakan kunci API untuk mengidentifikasi klien API dan siapa yang dapat mengakses tahapan API terkait untuk setiap kunci.

Kunci API adalah nilai string alfanumerik yang Anda distribusikan ke pelanggan pengembang aplikasi untuk memberikan akses ke API Anda. Anda dapat menggunakan kunci API bersama dengan [otorisasi Lambda](#), [peran IAM](#), atau Amazon Cognito untuk mengontrol [akses](#) ke Anda. APIs API Gateway dapat menghasilkan kunci API atas nama Anda, atau Anda dapat mengimpornya dari [file CSV](#). Anda dapat membuat kunci API di API Gateway, atau mengimpornya ke API Gateway dari sumber eksternal. Untuk informasi selengkapnya, lihat [the section called “Siapkan kunci API untuk REST APIs di API Gateway”](#).

Kunci API memiliki nama dan nilai. (Istilah “kunci API” dan “nilai kunci API” sering digunakan secara bergantian.) Nama tidak boleh melebihi 1024 karakter. Nilainya adalah string alfanumerik antara 20 dan 128 karakter, misalnya, `.apikey1234abcdefghij0123456789`

 Important

Nilai kunci API harus unik. Jika Anda mencoba membuat dua kunci API dengan nama berbeda dan nilai yang sama, API Gateway menganggapnya sebagai kunci API yang sama. Kunci API dapat dikaitkan dengan lebih dari satu paket penggunaan. Rencana penggunaan dapat dikaitkan dengan lebih dari satu tahap. Namun, kunci API tertentu hanya dapat dikaitkan dengan satu paket penggunaan untuk setiap tahap API Anda.

Batas pelambatan menetapkan titik target di mana pelambatan permintaan harus dimulai. Ini dapat diatur pada tingkat metode API atau API.

Batas kuota menetapkan jumlah maksimum target permintaan dengan kunci API tertentu yang dapat dikirimkan dalam interval waktu tertentu. Anda dapat mengonfigurasi metode API individual agar memerlukan otorisasi kunci API berdasarkan konfigurasi paket penggunaan.

Batas pembatasan dan kuota berlaku untuk permintaan kunci API individual yang digabungkan di semua tahapan API dalam paket penggunaan.

 Note

Pelambatan rencana penggunaan dan kuota bukanlah batas yang sulit, dan diterapkan atas dasar upaya terbaik. Dalam beberapa kasus, klien dapat melebihi kuota yang Anda tetapkan. Jangan mengandalkan kuota paket penggunaan atau pembatasan untuk mengontrol biaya atau memblokir akses ke API. Pertimbangkan [AWS Budgets](#) untuk menggunakan untuk memantau biaya dan [AWS WAF](#) mengelola permintaan API.

Praktik terbaik untuk kunci API dan paket penggunaan

Berikut ini adalah praktik terbaik yang disarankan untuk diikuti saat menggunakan kunci API dan rencana penggunaan.

 Important

- Jangan gunakan kunci API untuk otentikasi atau otorisasi untuk mengontrol akses ke Anda. APIs Jika Anda memiliki beberapa APIs dalam paket penggunaan, pengguna dengan kunci API yang valid untuk satu API dalam paket penggunaan tersebut dapat mengakses semua

APIs dalam paket penggunaan tersebut. Sebagai gantinya, untuk mengontrol akses ke API Anda, gunakan peran IAM, otorisasi [Lambda](#), atau kumpulan pengguna Amazon [Cognito](#).

- Gunakan kunci API yang dihasilkan API Gateway. Kunci API tidak boleh menyertakan informasi rahasia; klien biasanya mengirimkannya dalam header yang dapat dicatat.
- Jika Anda menggunakan portal pengembang untuk mempublikasikan APIs, perhatikan bahwa semua paket penggunaan Anda APIs dapat berlangganan oleh pelanggan, bahkan jika Anda belum membuatnya terlihat oleh pelanggan Anda.
- Dalam beberapa kasus, klien dapat melebihi kuota yang Anda tetapkan. Jangan mengandalkan rencana penggunaan untuk mengontrol biaya. Pertimbangkan [AWS Budgets](#) untuk menggunakan untuk memantau biaya dan [AWS WAF](#) mengelola permintaan API.
- Setelah Anda menambahkan kunci API ke paket penggunaan, operasi pembaruan mungkin memerlukan beberapa menit untuk menyelesaikannya.

Pilih sumber kunci API di API Gateway

Saat Anda mengaitkan rencana penggunaan dengan API dan mengaktifkan kunci API pada metode API, setiap permintaan masuk ke API harus berisi [kunci API](#). API Gateway membaca kunci dan membandingkannya dengan kunci dalam paket penggunaan. Jika ada kecocokan, API Gateway membatasi permintaan berdasarkan batas permintaan dan kuota paket. Kalau tidak, itu melempar `InvalidKeyParameter` pengecualian. Akibatnya, penelepon menerima 403 Forbidden respons.

API Gateway API Anda dapat menerima kunci API dari salah satu dari dua sumber:

HEADER

Anda mendistribusikan kunci API ke pelanggan Anda dan meminta mereka untuk meneruskan kunci API sebagai X-API-Key header dari setiap permintaan yang masuk.

AUTHORIZER

Anda memiliki otorisasi Lambda yang mengembalikan kunci API sebagai bagian dari respons otorisasi. Untuk informasi selengkapnya tentang respons otorisasi, lihat [the section called “Keluaran dari otorisasi API Gateway Lambda”](#).

 Note

Untuk praktik terbaik untuk dipertimbangkan, lihat [the section called “Praktik terbaik untuk kunci API dan paket penggunaan”](#).

Prosedur berikut menunjukkan cara memilih sumber kunci API untuk API.

Konsol Manajemen AWS

Untuk memilih sumber kunci API untuk API

1. Masuk ke konsol API Gateway.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih pengaturan API.
4. Di bagian detail API, pilih Edit.
5. Di bawah sumber kunci API, pilih Header atau Authorizer dari daftar tarik-turun.
6. Pilih Simpan perubahan.

AWS CLI

[update-rest-api](#) Perintah berikut memperbarui API untuk menyetel sumber kunci API keAUTHORIZER:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations  
op=replace,path=/apiKeySource,value=AUTHORIZER
```

Agar klien mengirimkan kunci API, atur value ke HEADER dalam perintah sebelumnya.

REST API

Untuk memilih sumber kunci API untuk API dengan menggunakan API Gateway REST API, panggil [restapi:update](#) sebagai berikut:

```
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"
    }
  ]
}

```

Agar otorisasi mengembalikan kunci API, setel value ke AUTHORIZER dalam patchOperations input sebelumnya.

Format file kunci API untuk API Gateway

API Gateway dapat mengimpor kunci API dari file eksternal dengan format nilai dipisahkan koma (CSV), lalu mengaitkan kunci yang diimpor dengan satu atau beberapa paket penggunaan. File yang diimpor harus berisi Key kolom Name dan. Nama header kolom tidak peka huruf besar/kecil, dan kolom dapat dalam urutan apa pun, seperti yang ditunjukkan pada contoh berikut:

```

Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey

```

KeyNilai harus antara 20 dan 128 karakter. NameNilai tidak boleh melebihi 1024 karakter.

File kunci API juga dapat memilikiDescription,Enabled, atau UsagePlanIds kolom, seperti yang ditunjukkan pada contoh berikut:

```

Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b

```

Ketika kunci dikaitkan dengan lebih dari satu paket penggunaan, UsagePlanIds nilainya adalah string yang dipisahkan koma dari rencana penggunaan IDs, diapit dengan sepasang tanda kutip ganda atau tunggal, seperti yang ditunjukkan pada contoh berikut:

```

Enabled,Name,key,UsageplanIds

```

```
true,MyFirstApiKey,apikey1234abcdefg hij0123456789,"c7y23b,glvrsr"
```

Kolom yang tidak dikenal diizinkan, tetapi diabaikan. Nilai default adalah string kosong atau nilai `true` Boolean.

Kunci API yang sama dapat diimpor beberapa kali, dengan versi terbaru menimpa yang sebelumnya. Dua kunci API identik jika memiliki key nilai yang sama.

Note

Untuk praktik terbaik untuk dipertimbangkan, lihat [the section called “Praktik terbaik untuk kunci API dan paket penggunaan”](#).

Siapkan kunci API untuk REST APIs di API Gateway

Untuk menyiapkan kunci API, lakukan hal berikut:

- Konfigurasi metode API untuk memerlukan kunci API.
- Membuat atau mengimpor kunci API untuk API di Wilayah.

Sebelum menyiapkan kunci API, Anda harus membuat API dan menerapkannya ke panggung. Setelah Anda membuat nilai kunci API, nilai tersebut tidak dapat diubah.

Untuk petunjuk tentang cara membuat dan menerapkan API menggunakan konsol API Gateway, lihat [Kembangkan REST API di API Gateway](#) dan [Menerapkan REST APIs di API Gateway](#), masing-masing.

Setelah membuat kunci API, Anda harus mengaitkannya dengan paket penggunaan. Untuk informasi selengkapnya, lihat [Siapkan paket penggunaan untuk REST APIs di API Gateway](#).

Note

Untuk praktik terbaik untuk dipertimbangkan, lihat [the section called “Praktik terbaik untuk kunci API dan paket penggunaan”](#).

Topik

- [Memerlukan kunci API pada suatu metode](#)
- [Membuat kunci API](#)
- [Impor kunci API](#)

Memerlukan kunci API pada suatu metode

Prosedur berikut menjelaskan cara mengonfigurasi metode API agar memerlukan kunci API.

Konsol Manajemen AWS

Untuk mengonfigurasi metode API agar memerlukan kunci API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama API Gateway, pilih Resources.
4. Di bawah Sumber Daya, buat metode baru atau pilih yang sudah ada.
5. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.

The screenshot shows the Amazon API Gateway console interface. On the left, a sidebar displays a resource tree with the path `/pets` and the method `GET` selected. The main panel is titled 'Method request settings' and contains several configuration sections:

- Authorization:** Set to `NONE`.
- Request validator:** Set to `None`.
- API key required:** A checkbox labeled 'API key required' is set to `False`. This checkbox is highlighted with a red rectangular box, and a blue 'Edit' button is visible next to it.
- SDK operation name:** Set to 'Generated based on method and path'.
- Request paths (0):** A section indicating no request paths are currently defined.
- URL query string parameters (2):** A table listing two parameters: 'page' and 'type', both with 'Required' set to `False` and 'Caching' set to `Inactive`.
- HTTP request headers (2):** A table listing one header: 'header1', with 'Required' set to `False` and 'Caching' set to `Inactive`.

6. Pilih kunci API yang diperlukan.
7. Pilih Simpan.
8. Menerapkan atau menerapkan ulang API agar persyaratan diterapkan.

Jika opsi yang diperlukan kunci API disetel ke `false` dan Anda tidak menjalankan langkah sebelumnya, kunci API apa pun yang terkait dengan tahap API tidak akan digunakan untuk metode tersebut.

AWS CLI

Perintah [put-method](#) berikut membuat PUT metode yang memerlukan kunci API:

```
aws apigateway put-method \
  --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method PUT \
```

```
--authorization-type "NONE" \  
--api-key-required
```

Perintah [update-method](#) berikut memperbarui metode yang ada untuk memerlukan kunci API:

```
aws apigateway update-method \  
  --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --patch-operations op="replace",path="/apiKeyRequired",value="true"
```

REST API

Untuk mewajibkan kunci API pada suatu metode, lakukan salah satu hal berikut:

- Panggilan [method:put](#) untuk membuat metode. Setel `apiKeyRequired` ke `true` dalam payload permintaan.
- Panggilan [method:update](#) untuk mengatur `apiKeyRequired` ke `true`.

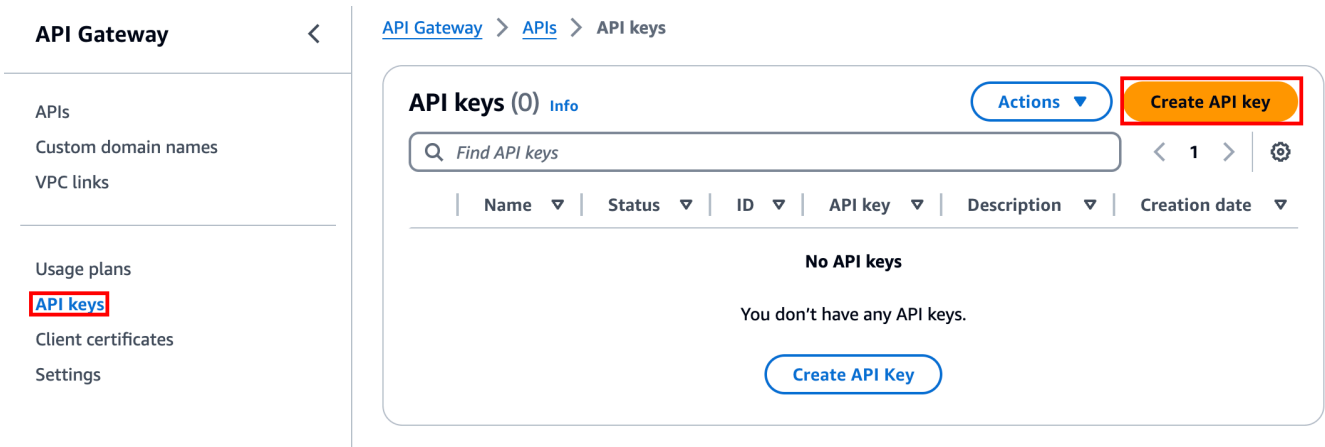
Membuat kunci API

Prosedur berikut menunjukkan cara membuat kunci API. Jika Anda ingin mengimpor kunci API Anda, lewati langkah ini.

Konsol Manajemen AWS

Untuk membuat kunci API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama API Gateway, pilih kunci API.
4. Pilih Buat kunci API.



5. Untuk Nama, masukkan nama.
6. (Opsional) Untuk Deskripsi, masukkan deskripsi.
7. Untuk kunci API, pilih Auto generate agar API Gateway menghasilkan nilai kunci, atau pilih Custom untuk membuat nilai kunci Anda sendiri.
8. Pilih Simpan.

AWS CLI

[create-api-key](#) Perintah berikut membuat kunci API:

```
aws apigateway create-api-key \
  --name 'Dev API key' \
  --description 'API key for Devs' \
  --enabled
```

REST API

Panggilan [apikey:create](#) untuk membuat kunci API.

Impor kunci API

Prosedur berikut menjelaskan cara mengimpor kunci API. Jika Anda sudah membuat kunci API, lewati langkah ini.

Konsol Manajemen AWS

Untuk mengimpor kunci API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih kunci API.
4. Pilih menu tarik-turun Tindakan, lalu pilih Impor kunci API.
5. Untuk memuat file kunci yang dipisahkan koma, pilih Pilih file. Anda juga dapat memasukkan tombol di editor teks. Untuk informasi tentang format file, lihat [the section called “Format file kunci API untuk API Gateway”](#).
6. Pilih Gagal pada peringatan untuk menghentikan impor ketika ada kesalahan, atau pilih Abaikan peringatan untuk terus mengimpor entri kunci yang valid saat ada peringatan.
7. Pilih Impor untuk mengimpor kunci API Anda.

AWS CLI

[import-api-keys](#) Perintah berikut mengimpor kunci API:

```
aws apigateway import-api-key \  
  a--body fileb://keys.csv \  
  --format csv
```

REST API

Panggilan [apikey:import](#) untuk mengimpor kunci API dari file. Untuk format file, lihat [Format file kunci API untuk API Gateway](#).

Anda tidak dapat mengubah nilai kunci API baru. Setelah membuat API, Anda mengonfigurasi paket penggunaan. Untuk informasi selengkapnya, lihat [Siapkan paket penggunaan untuk REST APIs di API Gateway](#).

Siapkan paket penggunaan untuk REST APIs di API Gateway

Sebelum membuat paket penggunaan, pastikan Anda telah menyiapkan kunci API Anda. Untuk informasi selengkapnya, lihat [Siapkan kunci API untuk REST APIs di API Gateway](#).

Topik

- [Migrasi API Anda ke paket penggunaan default \(jika diperlukan\)](#)
- [Buat rencana penggunaan](#)
- [Tambahkan tahapan ke paket penggunaan](#)
- [Menambahkan kunci API ke paket penggunaan](#)

Migrasi API Anda ke paket penggunaan default (jika diperlukan)

Jika Anda mulai menggunakan API Gateway setelah fitur paket penggunaan diluncurkan pada 11 Agustus 2016, Anda akan secara otomatis mengaktifkan paket penggunaan untuk Anda di semua Wilayah yang didukung.

Jika Anda mulai menggunakan API Gateway sebelum tanggal tersebut, Anda mungkin perlu bermigrasi ke paket penggunaan default. Anda akan diminta dengan opsi Aktifkan Paket Penggunaan sebelum menggunakan paket penggunaan untuk pertama kalinya di Wilayah yang dipilih. Saat mengaktifkan opsi ini, Anda memiliki paket penggunaan default yang dibuat untuk setiap tahap API unik yang terkait dengan kunci API yang ada. Dalam paket penggunaan default, tidak ada batas throttle atau kuota yang ditetapkan pada awalnya, dan asosiasi antara kunci API dan tahapan API disalin ke paket penggunaan. API berperilaku sama seperti sebelumnya. Namun, Anda harus menggunakan [UsagePlan](#)`apiStages`property untuk mengaitkan nilai tahap API tertentu (`apiId`dan `stage`) dengan kunci API yang disertakan (via [UsagePlanKey](#)), alih-alih menggunakan [ApiKey](#)`stageKeys`property.

Untuk memeriksa apakah Anda sudah bermigrasi ke paket penggunaan default, gunakan perintah [get-account](#)CLI. Dalam output perintah, `features` daftar menyertakan entri "UsagePlans" kapan rencana penggunaan diaktifkan.

Anda juga dapat memigrasikan APIs ke paket penggunaan default dengan menggunakan AWS CLI sebagai berikut:

Untuk bermigrasi ke paket penggunaan default menggunakan AWS CLI

1. Panggil perintah CLI ini: [update-account](#)
2. Untuk `cli-input-json` parameter, gunakan JSON berikut:

```
[  
  {
```

```
    "op": "add",  
    "path": "/features",  
    "value": "UsagePlans"  
  }  
]
```

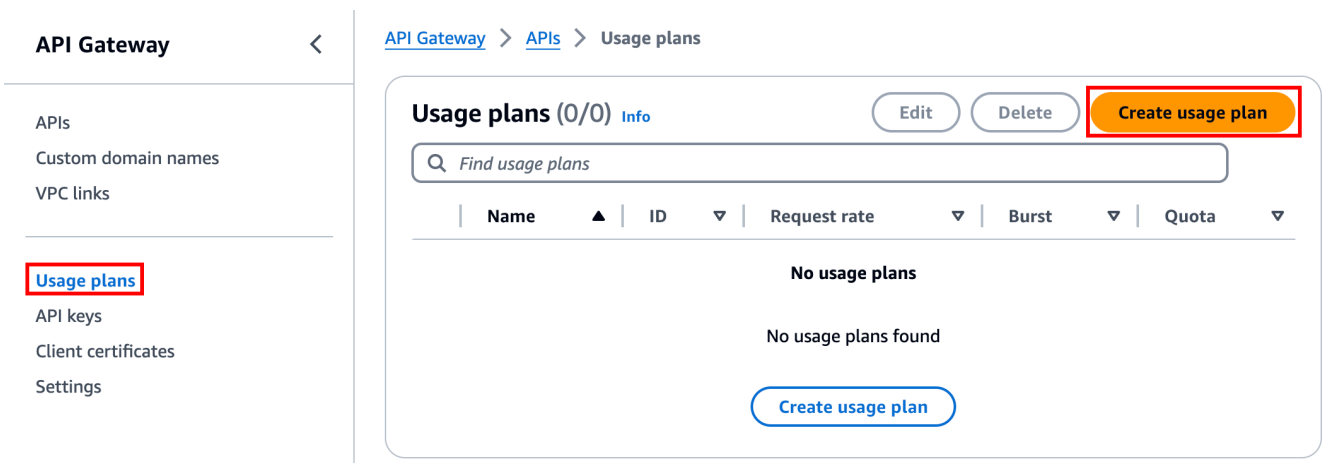
Buat rencana penggunaan

Prosedur berikut menjelaskan cara membuat rencana penggunaan.

Konsol Manajemen AWS

Untuk membuat rencana penggunaan

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan, lalu pilih Buat paket penggunaan.



3. Untuk Nama, masukkan nama.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Secara default, paket penggunaan mengaktifkan pelambatan. Masukkan Rate dan Burst untuk paket penggunaan Anda. Pilih Throttling untuk mematikan throttling.
6. Secara default, paket penggunaan mengaktifkan kuota untuk jangka waktu tertentu. Untuk Permintaan, masukkan jumlah total permintaan yang dapat dibuat pengguna dalam periode waktu paket penggunaan Anda. Pilih Kuota untuk mematikan kuota.
7. Pilih Buat paket penggunaan.

AWS CLI

[create-usage-plan](#) Perintah berikut membuat paket penggunaan yang mengatur ulang pada awal bulan:

```
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

Hubungi [usageplan:create](#) untuk membuat paket penggunaan.

Tambahkan tahapan ke paket penggunaan

Prosedur berikut menjelaskan cara menambahkan tahapan ke rencana penggunaan.

Konsol Manajemen AWS

Untuk menambahkan tahapan ke rencana penggunaan

1. Pilih paket penggunaan Anda.
2. Di bawah tab Tahapan terkait, pilih Tambahkan tahap.

[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. Untuk API, pilih API.
4. Untuk Stage, pilih panggung.
5. (Opsional) Untuk mengaktifkan pelambatan tingkat metode, lakukan hal berikut:
 - a. Pilih pelambatan tingkat metode, lalu pilih Tambahkan metode.
 - b. Untuk Sumber Daya, pilih sumber daya dari API Anda.
 - c. Untuk Metode, pilih metode dari API Anda.
 - d. Masukkan Rate dan Burst untuk paket penggunaan Anda.
6. Pilih Tambahkan ke paket penggunaan.

AWS CLI

[update-usage-plan](#) Perintah berikut menambahkan Prod tahap API ke paket penggunaan:

```
aws apigateway update-usage-plan \  
  --usage-plan-id abc123 \  
  --patch-operations op="add",path="/apiStages",value="a1b1c2:Prod"
```

REST API

Hubungi [usageplan:update](#) untuk memperbarui paket penggunaan.

Menambahkan kunci API ke paket penggunaan

Prosedur berikut menunjukkan cara menambahkan kunci API ke paket penggunaan.

Konsol Manajemen AWS

Untuk menambahkan kunci ke paket penggunaan

1. Di bawah tab Kunci API Terkait, pilih Tambahkan kunci API.

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. Untuk mengaitkan kunci yang ada ke paket penggunaan Anda, pilih Tambahkan kunci yang ada, lalu pilih kunci yang ada dari menu tarik-turun.
b. Untuk membuat kunci API baru, pilih Buat dan tambahkan kunci baru, lalu buat kunci baru. Untuk informasi selengkapnya tentang cara membuat kunci baru, lihat [Membuat kunci API](#).
3. Pilih Tambahkan kunci API.

AWS CLI

[create-usage-plan-key](#) Perintah berikut mengaitkan kunci API yang ada dengan paket penggunaan:

```
aws apigateway create-usage-plan-key \
  --usage-plan-id a1b2c3 \
  --key-type "API_KEY" \
  --key-id aaa111bbb
```

REST API

Panggilan [usageplankey:create](#) untuk mengaitkan kunci API yang ada dengan paket penggunaan.

Anda juga dapat langsung mengaitkan APIs kunci Anda dengan paket penggunaan saat Anda mengimpornya. Panggil [apikey:import](#) untuk menambahkan satu atau beberapa kunci API langsung ke paket penggunaan yang ditentukan. Payload permintaan harus berisi nilai kunci API, pengenal paket penggunaan terkait, flag Boolean untuk menunjukkan bahwa kunci diaktifkan untuk paket penggunaan, dan, mungkin, nama dan deskripsi kunci API.

Contoh `apikey:import` permintaan berikut ini menambahkan tiga kunci API (seperti yang diidentifikasi oleh `key`, `name`, dan `description`) ke satu paket penggunaan (seperti yang diidentifikasi oleh `usageplanIds`):

```
POST /apikeys?mode=import&format=csv&failonwarnings=fase 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
```

Akibatnya, tiga `UsagePlanKey` sumber daya dibuat dan ditambahkan ke `UsagePlan`.

Anda juga dapat menambahkan kunci API ke lebih dari satu paket penggunaan dengan cara ini. Untuk melakukan ini, ubah setiap nilai `usageplanIds` kolom menjadi string yang dipisahkan koma yang berisi pengidentifikasi rencana penggunaan yang dipilih, dan diapit dalam sepasang tanda kutip (atau). `"n371pt,m282qs" 'n371pt,m282qs'`

Note

Kunci API dapat dikaitkan dengan lebih dari satu paket penggunaan. Rencana penggunaan dapat dikaitkan dengan lebih dari satu tahap. Namun, kunci API tertentu hanya dapat dikaitkan dengan satu paket penggunaan untuk setiap tahap API Anda.

Mempertahankan rencana penggunaan untuk REST APIs di API Gateway

Mempertahankan rencana penggunaan melibatkan pemantauan kuota yang digunakan dan yang tersisa selama periode waktu tertentu dan, jika diperlukan, memperpanjang kuota yang tersisa dengan jumlah tertentu. Prosedur berikut menjelaskan cara memantau kuota.

Konsol Manajemen AWS

Untuk memantau kuota yang digunakan dan yang tersisa

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan.
3. Pilih paket penggunaan.
4. Pilih tab Kunci API Terkait untuk melihat jumlah permintaan yang tersisa untuk periode waktu untuk setiap kunci.
5. (Opsional) Pilih Ekspor data penggunaan, lalu pilih Tanggal Dari dan Tanggal Sampai saat ini. Kemudian pilih JSON atau CSV untuk format data yang diekspor, lalu pilih Ekspor.

Contoh berikut menunjukkan file yang diekspor.

```
{
  "px1KW6...qBaz0JH": [
    [
      0,
      5000
    ],
    [
      0,
      5000
    ],
    [
      0,
```

```
    10  
  ]  
]  
}
```

Data penggunaan dalam contoh menunjukkan data penggunaan harian untuk klien API, seperti yang diidentifikasi oleh kunci API (px1KW6 . . . qBaz0JH), antara 1 Agustus 2016 dan 3 Agustus 2016. Setiap data penggunaan harian menunjukkan kuota yang digunakan dan yang tersisa. Dalam contoh ini, pelanggan belum menggunakan kuota yang dialokasikan, dan pemilik atau administrator API telah mengurangi sisa kuota dari 5000 menjadi 10 pada hari ketiga.

Prosedur berikut menjelaskan cara memodifikasi kuota.

Untuk memperpanjang kuota yang tersisa

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan.
3. Pilih paket penggunaan.
4. Pilih tab Kunci API Terkait untuk melihat jumlah permintaan yang tersisa untuk periode waktu untuk setiap kunci.
5. Pilih kunci API, lalu pilih Grant usage extension.
6. Masukkan nomor untuk kuota Permintaan yang tersisa. Anda dapat meningkatkan permintaan penggantian nama atau mengurangi permintaan yang tersisa untuk jangka waktu paket penggunaan Anda.
7. Pilih Perbarui kuota.

AWS CLI

[update-usage-plan](#) Contoh berikut menambah, menghapus, atau memodifikasi pengaturan pelambatan tingkat metode dalam paket penggunaan.

Note

Pastikan untuk mengubah us-east-1 ke nilai Region yang sesuai untuk API Anda.

Untuk menambah atau mengganti batas tarif untuk membatasi sumber daya dan metode individual:

```
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"
```

Untuk menambah atau mengganti batas burst untuk membatasi sumber daya dan metode individual:

```
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"
```

Untuk menghapus pengaturan pelambatan tingkat metode untuk sumber daya dan metode individual:

```
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=""
```

Untuk menghapus semua setelan pelambatan tingkat metode untuk API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="remove",path="/apiStages/apiId:stage/throttle ",value=""
```

Berikut adalah contoh menggunakan contoh API Pet Store:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="replace",path="/apiStages/apiId:stage/throttle",value='{"\n/pets/GET\n":{"rateLimit\n":1.0,\n"burstLimit\n":1},\n"/GET\n":{"rateLimit\n":1.0,\n"burstLimit\n":1}}'
```

REST API

Hubungi [usageplan:update](#) untuk mempertahankan rencana penggunaan.

Membuat dan mengonfigurasi kunci API dan rencana penggunaan dengan CloudFormation

Anda dapat menggunakan CloudFormation untuk mewajibkan kunci API pada metode API dan membuat rencana penggunaan untuk API. Contoh CloudFormation template melakukan hal berikut:

- Membuat API Gateway API dengan GET dan POST metode.
- Memerlukan kunci API untuk GET dan POST metode. API ini menerima kunci dari X-API-KEY header setiap permintaan yang masuk.
- Membuat kunci API.
- Membuat paket penggunaan untuk menentukan kuota bulanan 1.000 permintaan setiap bulan, batas laju pelambatan 100 permintaan setiap detik, dan batas burst throttling 200 permintaan setiap detik.
- Menentukan batas laju pelambatan tingkat metode 50 permintaan setiap detik dan batas burst throttling tingkat metode 100 permintaan per detik untuk metode tersebut. GET
- Mengaitkan tahap API dan kunci API dengan paket penggunaan.

```
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}'

```

Konfigurasi metode untuk menggunakan kunci API dengan definisi OpenAPI

Anda dapat menggunakan definisi OpenAPI untuk meminta kunci API pada suatu metode.

Untuk setiap metode, buat objek persyaratan keamanan untuk meminta kunci API untuk menjalankan metode tersebut. Kemudian, tentukan `api_key` dalam definisi keamanan. Setelah membuat API, tambahkan tahap API baru ke paket penggunaan Anda.

Contoh berikut membuat API dan memerlukan kunci API untuk POST dan GET metode:

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"
    }
  }
}
```

```

    }
  }
}

```

Uji paket penggunaan untuk REST APIs di API Gateway

Sebagai contoh, mari kita gunakan PetStore API, yang dibuat di [Tutorial: Buat REST API dengan mengimpor contoh](#). Asumsikan bahwa API dikonfigurasi untuk menggunakan kunci API dari `Hiorr45VR...c4GJc`. Langkah-langkah berikut menjelaskan cara menguji rencana penggunaan.

Untuk menguji paket penggunaan Anda

- Buat GET permintaan pada resource Pets (`/pets`), dengan parameter `?type=...&page=...` kueri, API (misalnya, `xbvxlpjch`) dalam paket penggunaan:

```

GET /testStage/pets?type=dog&page=1 HTTP/1.1
x-api-key: Hiorr45VR...c4GJc
Content-Type: application/x-www-form-urlencoded
Host: xbvxlpjch.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

Anda harus mengirimkan permintaan ini ke `execute-api` komponen API Gateway dan memberikan kunci API yang diperlukan (misalnya, `Hiorr45VR...c4GJc`) di `x-api-key` header yang diperlukan.

Respons yang berhasil mengembalikan kode `200 OK` status dan payload yang berisi hasil yang diminta dari backend. Jika Anda lupa mengatur `x-api-key` header atau mengaturnya dengan kunci yang salah, Anda mendapatkan `403 Forbidden` respons. Namun, jika Anda tidak mengonfigurasi metode untuk memerlukan kunci API, kemungkinan besar Anda akan

mendapatkan 200 OK respons apakah Anda menyetel `x-api-key` header dengan benar atau tidak, dan batas throttle dan kuota paket penggunaan dilewati.

Kadang-kadang, ketika terjadi kesalahan internal di mana API Gateway tidak dapat menerapkan batas pembatasan rencana penggunaan atau kuota untuk permintaan, API Gateway melayani permintaan tanpa menerapkan batas pembatasan atau kuota seperti yang ditentukan dalam paket penggunaan. Tapi, itu mencatat pesan kesalahan `Usage Plan check failed due to an internal error in CloudWatch`. Anda dapat mengabaikan kesalahan sesekali seperti itu.

Panggil metode menggunakan kunci API

Bergantung pada jenis sumber kunci API yang Anda pilih, gunakan salah satu prosedur berikut untuk menggunakan kunci API bersumber header atau kunci API yang dikembalikan oleh otorisasi dalam pemanggilan metode:

Untuk menggunakan kunci API bersumber header:

1. Buat API dengan metode API yang diinginkan, lalu terapkan API ke panggung.
2. Buat paket penggunaan baru atau pilih yang sudah ada. Tambahkan tahap API yang diterapkan ke paket penggunaan. Lampirkan kunci API ke paket penggunaan atau pilih kunci API yang ada dalam paket. Perhatikan nilai kunci API yang dipilih.
3. Siapkan metode API untuk memerlukan kunci API.
4. Menerapkan ulang API ke tahap yang sama. Jika Anda menerapkan API ke tahap baru, pastikan untuk memperbarui paket penggunaan untuk melampirkan tahap API baru.
5. Panggil API menggunakan kunci API. Contoh perintah curl berikut memanggil GET metode pada `getUsers` sumber daya prod tahap API menggunakan kunci API.

```
curl -H "X-API-Key: abcd1234" 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Klien sekarang dapat memanggil metode API sambil memasok `x-api-key` header dengan kunci API yang dipilih sebagai nilai header. Panggilan mungkin terlihat seperti berikut:

Untuk menggunakan kunci API yang bersumber dari otorisasi:

1. Buat API dengan metode API yang diinginkan, lalu terapkan API ke panggung.

2. Buat paket penggunaan baru atau pilih yang sudah ada. Tambahkan tahap API yang diterapkan ke paket penggunaan. Lampirkan kunci API ke paket penggunaan atau pilih kunci API yang ada dalam paket. Perhatikan nilai kunci API yang dipilih.
3. Buat otorisasi Lambda berbasis token. Sertakan, `usageIdentifierKey:{api-key}` sebagai properti tingkat root dari respons otorisasi. Untuk petunjuk cara membuat otorisasi berbasis token, lihat [the section called “Contoh fungsi Lambda TOKEN authorizer”](#)
4. Siapkan metode API untuk memerlukan kunci API dan aktifkan otorisasi Lambda pada metode juga.
5. Menerapkan ulang API ke tahap yang sama. Jika Anda menerapkan API ke tahap baru, pastikan untuk memperbarui paket penggunaan untuk melampirkan tahap API baru.

Klien sekarang dapat memanggil metode yang diperlukan kunci API tanpa secara eksplisit memasok kunci API apa pun. Kunci API yang dikembalikan authorizer digunakan secara otomatis.

Dokumentasi untuk REST APIs di API Gateway

Untuk membantu pelanggan memahami dan menggunakan API Anda, Anda harus mendokumentasikan API. Untuk membantu Anda mendokumentasikan API, API Gateway memungkinkan Anda menambahkan dan memperbarui konten bantuan untuk entitas API individual sebagai bagian integral dari proses pengembangan API Anda. API Gateway menyimpan konten sumber dan memungkinkan Anda mengarsipkan berbagai versi dokumentasi. Anda dapat mengaitkan versi dokumentasi dengan tahap API, mengekspor snapshot dokumentasi khusus tahap ke file OpenAPI eksternal, dan mendistribusikan file sebagai publikasi dokumentasi.

Untuk mendokumentasikan API Anda, Anda dapat memanggil [API Gateway REST API](#), menggunakan salah satunya [AWS SDKs](#), menggunakan [AWS CLI](#) for API Gateway, atau menggunakan konsol API Gateway. Selain itu, Anda dapat mengimpor atau mengekspor bagian dokumentasi yang didefinisikan dalam file OpenAPI eksternal.

Untuk berbagi dokumentasi API dengan pengembang, Anda dapat menggunakan portal pengembang. Sebagai contoh, lihat [Mengintegrasikan ReadMe dengan API Gateway untuk Menjaga Hub Pengembang Anda Tetap Terbaru atau Cara Merampingkan Pengembangan API di Amazon API Gateway Menggunakan blog SmartBear](#) Jaringan AWS Partner (APN). SwaggerHub

Topik

- [Representasi dokumentasi API di API Gateway](#)
- [Dokumentasikan API menggunakan konsol API Gateway](#)

- [Publikasikan dokumentasi API menggunakan konsol API Gateway](#)
- [Dokumentasikan API menggunakan API Gateway REST API](#)
- [Publikasikan dokumentasi API menggunakan API Gateway REST API](#)
- [Impor dokumentasi API](#)
- [Kontrol akses ke dokumentasi API di API Gateway](#)

Representasi dokumentasi API di API Gateway

Dokumentasi API Gateway API terdiri dari bagian-bagian dokumentasi individual yang terkait dengan entitas API tertentu yang mencakup API, sumber daya, metode, permintaan, respons, parameter pesan (yaitu, jalur, kueri, header), serta otorisasi dan model.

Di API Gateway, bagian dokumentasi diwakili oleh [DocumentationPart](#) sumber daya. Dokumentasi API secara keseluruhan diwakili oleh [DocumentationParts](#) koleksi.

Mendokumentasikan API melibatkan pembuatan `DocumentationPart` instance, menambahkannya ke `DocumentationParts` koleksi, dan memelihara versi bagian dokumentasi saat API Anda berkembang.

Topik

- [Bagian dokumentasi](#)
- [Versi dokumentasi](#)

Bagian dokumentasi

[DocumentationPart](#) Sumber daya adalah objek JSON yang menyimpan konten dokumentasi yang berlaku untuk entitas API individual. Ini termasuk konten UTF-8 dan semua bahasa pelokalan utama untuk dokumentasi Anda. `properties` Bidanganya berisi konten dokumentasi sebagai peta pasangan kunci-nilai. `location` Properti mengidentifikasi entitas API terkait.

Bentuk peta konten ditentukan oleh Anda, pengembang API. Nilai pasangan kunci-nilai dapat berupa string, angka, boolean, objek, atau array. Bentuk `location` objek tergantung pada jenis entitas yang ditargetkan.

`DocumentationPart` Sumber daya mendukung pewarisan konten: konten dokumentasi entitas API berlaku untuk turunan dari entitas API tersebut. Untuk informasi selengkapnya tentang definisi entitas turunan dan pewarisan konten, lihat [Mewarisi Konten dari Entitas API dengan Spesifikasi Lebih Umum](#).

Lokasi bagian dokumentasi

Properti [lokasi DocumentationPart](#) instance mengidentifikasi entitas API yang menerapkan konten terkait. Entitas API dapat berupa sumber API API Gateway REST API, seperti, [Resource RestApi](#), [Method](#), [MethodResponse](#), [Authorizer](#), atau [Model](#). Entitas juga dapat berupa parameter pesan, seperti parameter jalur URL, parameter string kueri, parameter header permintaan atau respons, badan permintaan atau respons, atau kode status respons.

Untuk menentukan entitas API, tetapkan atribut [type](#) location objek menjadi salah satu dari API, AUTHORIZER, MODEL, RESOURCE, METHODPATH_PARAMETER, QUERY_PARAMETER, REQUEST_HEADER, REQUEST_BODY, RESPONSE, RESPONSE_BODY, atau RESPONSE_BODY.

Bergantung pada type entitas API, Anda dapat menentukan location atribut lain, termasuk [metode](#), [nama](#), [jalur](#), dan [StatusCode](#). Tidak semua atribut ini valid untuk entitas API tertentu. Misalnya, type, pathname, dan statusCode merupakan atribut valid dari RESPONSE entitas; hanya type dan path merupakan atribut lokasi yang valid dari RESOURCE entitas. Merupakan kesalahan untuk menyertakan bidang yang tidak valid di location a DocumentationPart untuk entitas API tertentu.

Tidak semua location bidang yang valid diperlukan. Misalnya, type adalah location bidang yang valid dan wajib dari semua entitas API. Namun, method, path, dan statusCode merupakan atribut yang valid tetapi tidak diperlukan untuk RESPONSE entitas. Ketika tidak ditentukan secara eksplisit, location bidang yang valid mengasumsikan nilai defaultnya. pathNilai defaultnya adalah /, yaitu sumber daya root dari API. Nilai defaultmethod, atau statusCode is*, yang berarti metode apa pun, atau nilai kode status, masing-masing.

Isi bagian dokumentasi

propertiesNilai dikodekan sebagai string JSON. propertiesNilai berisi informasi apa pun yang Anda pilih untuk memenuhi persyaratan dokumentasi Anda. Misalnya, berikut ini adalah peta konten yang valid:

```
{
  "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."
}
```

Meskipun API Gateway menerima string JSON yang valid sebagai peta konten, atribut konten diperlakukan sebagai dua kategori: atribut yang dapat dikenali oleh OpenAPI dan yang tidak bisa. Dalam contoh sebelumnya,, `infodescription`, dan `x-custom-info` diakui oleh OpenAPI sebagai objek OpenAPI standar, properti, atau ekstensi. Sebaliknya, tidak `my-info` sesuai dengan spesifikasi OpenAPI. API Gateway menyebarkan atribut konten yang sesuai dengan OpenAPI ke dalam definisi entitas API dari instance terkait. `DocumentationPart` API Gateway tidak menyebarkan atribut konten yang tidak sesuai ke dalam definisi entitas API.

Sebagai contoh lain, di sini `DocumentationPart` ditargetkan untuk `Resource` entitas:

```
{
  "location" : {
    "type" : "RESOURCE",
    "path": "/pets"
  },
  "properties" : {
    "summary" : "The /pets resource represents a collection of pets in PetStore.",
    "description": "... a child resource under the root...",
  }
}
```

Di sini, path keduanya `type` dan merupakan bidang yang valid untuk mengidentifikasi target `RESOURCE` jenis. Untuk sumber daya root (/), Anda dapat menghilangkan path bidang.

```
{
  "location" : {
    "type" : "RESOURCE"
  },
  "properties" : {
    "description" : "The root resource with the default path specification."
  }
}
```

Ini sama dengan `DocumentationPart` contoh berikut:

```
{
  "location" : {
    "type" : "RESOURCE",
    "path": "/"
  },
  "properties" : {
```

```

    "description" : "The root resource with an explicit path specification"
  }
}

```

Mewarisi konten dari entitas API dengan spesifikasi yang lebih umum

Nilai default `location` bidang opsional memberikan deskripsi berpola entitas API. Menggunakan nilai default dalam `location` objek, Anda dapat menambahkan deskripsi umum di `properties` peta ke `DocumentationPart` instance dengan jenis `location` pola ini. API Gateway mengekstrak atribut dokumentasi OpenAPI yang berlaku dari entitas API generik dan menyuntikkannya ke entitas API tertentu dengan `location` bidang yang cocok dengan `location` pola umum, atau mencocokkan nilai persisnya, kecuali entitas tertentu sudah memiliki `DocumentationPart` instance yang terkait dengannya. `DocumentationPart` Perilaku ini juga dikenal sebagai pewarisan konten dari entitas API dengan spesifikasi yang lebih umum.

Warisan konten tidak berlaku untuk tipe entitas API tertentu. Lihat tabel di bawah ini untuk detailnya.

Jika entitas API cocok dengan lebih dari pola lokasi `DocumentationPart` seseorang, entitas akan mewarisi bagian dokumentasi dengan bidang lokasi dengan prioritas dan kekhususan tertinggi. Urutan prioritas adalah `path > statusCode` Untuk mencocokkan dengan `path` bidang, API Gateway memilih entitas dengan nilai jalur paling spesifik. Tabel berikut menunjukkan ini dengan beberapa contoh.

Kasu	path	stat e	name	Ketera n
1	/pets	*	id	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi oleh entitas

Kasu	path	stat	name	Ketera	
		e		n	
				yang	
				cocok	
				dengan	
				pola	
				lokasi.	

Kasu	path	stat e	name	Ketera n
2	/pets	200	id	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi oleh entitas yang cocok dengan pola lokasi ketika Kasus 1 dan Kasus 2 dicocokka n, karena Kasus 2 lebih spesifik daripada Kasus 1.

Kasu	path	stat e	name	Ketera n
3	/pets/ petId	*	id	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi oleh entitas yang cocok dengan pola lokasi ketika Kasus 1, 2, dan 3 dicocokka n, karena Kasus 3 memiliki prioritas yang lebih tinggi daripada

Kasu	path	stat	name	Ketera	
		e		n	
				Kasus	
				2	
				dan	
				lebih	
				spesifik	
				daripada	
				Kasus	
				1.	

Berikut adalah contoh lain untuk membandingkan DocumentationPart contoh yang lebih umum dengan yang lebih spesifik. Pesan kesalahan umum berikut "Invalid request error" ini disuntikkan ke dalam definisi OpenAPI dari respons 400 kesalahan, kecuali diganti.

```
{
  "location" : {
    "type" : "RESPONSE",
    "statusCode": "400"
  },
  "properties" : {
    "description" : "Invalid request error."
  }
}
```

Dengan penempatan berikut, 400 tanggapan terhadap metode apa pun pada /pets sumber daya memiliki deskripsi sebagai "Invalid petId specified" gantinya.

```
{
  "location" : {
    "type" : "RESPONSE",
    "path": "/pets",
    "statusCode": "400"
  },
  "properties" : {
    "description" : "Invalid petId specified."
  }
}
```

Bidang lokasi yang valid dari **DocumentationPart**

Tabel berikut menunjukkan bidang yang valid dan wajib serta nilai default yang berlaku dari [DocumentationPart](#) sumber daya yang dikaitkan dengan jenis entitas API tertentu.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
API	<pre>{ "location": { "type": "API" }, ... }</pre>	type	N/A	Tidak
Sumber Daya	<pre>{ "location": { "type": "RESOURCE" }, "path": "<i>resource_path</i> " }, ... }</pre>	type	Nilai default path adalah /.	Tidak
Metode	<pre>{ "location": { "type": "METHOD", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> " }, ... }</pre>	type	Nilai default path dan method adalah / dan *, masing-masing.	Ya, cocok path dengan awalan dan pencocokan method nilai apa pun.
Parameter kueri	<pre>{</pre>	type	Nilai default path dan method	Ya, mencocokk

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
	<pre> "location": { "type": "QUERY_PARAMETER", "path": "<i>resource_path</i> ", "method": "<i>HTTP_verb</i> ", "name": "<i>query_parameter_name</i> " }, ... </pre>		adalah / dan*, masing-masing.	an path dengan awalan dan mencocokkan method dengan nilai yang tepat.
Isi permintaan	<pre> { "location": { "type": "REQUEST_BODY", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> " }, ... } </pre>	type	Nilai default dari path, dan method are / dan*, masing-masing.	Ya, mencocokkan path dengan awalan, dan mencocokkan method dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Permintaan parameter header	<pre>{ "location": { "type": "REQUEST_HEADER", "path": "<i>resource_path</i> ", "method": "<i>HTTP_verb</i> ", "name": "<i>header_name</i> " }, ... }</pre>	type, name	Nilai default path dan method adalah / dan*, masing-masing.	Ya, mencocokkan path dengan awalan dan mencocokkan method dengan nilai yang tepat.
Parameter jalur permintaan	<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	Nilai default path dan method adalah / dan*, masing-masing.	Ya, mencocokkan path dengan awalan dan mencocokkan method dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Respons	<pre>{ "location": { "type": "RESPONSE", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> ", "statusCode": "<i>status_code</i> " }, ... }</pre>	type	Nilai default dari path, method, dan statusCode adalah /, * dan *, masing-masing.	Ya, cocok path dengan awalan dan pencocokan method dan statusCode dengan nilai yang tepat.
Header respons	<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	Nilai default dari path, method dan statusCode adalah /, * dan *, masing-masing.	Ya, cocok path dengan awalan dan pencocokan method, dan statusCode dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Isi respons	<pre>{ "location": { "type": "RESPONSE_BODY", "path": "<i>resource_path</i> ", "method": "<i>http_verb</i> ", "statusCode": "<i>status_code</i> " }, ... }</pre>	type	Nilai default dari path, method dan statusCode are /, * dan *, masing-masing.	Ya, cocok path dengan awalan dan pencocokan method, dan statusCode dengan nilai yang tepat.
Pengotorisasi	<pre>{ "location": { "type": "AUTHORIZER", "name": "<i>authorizer_name</i> " }, ... }</pre>	type	N/A	Tidak
Model	<pre>{ "location": { "type": "MODEL", "name": "<i>model_name</i> " }, ... }</pre>	type	N/A	Tidak

Versi dokumentasi

Versi dokumentasi adalah snapshot dari [DocumentationParts](#) koleksi API dan ditandai dengan pengenal versi. Menerbitkan dokumentasi API melibatkan pembuatan versi dokumentasi, mengaitkannya dengan tahap API, dan mengeksport versi spesifik tahap dokumentasi API ke file OpenAPI eksternal. Di API Gateway, snapshot dokumentasi direpresentasikan sebagai [DocumentationVersion](#) sumber daya.

Saat memperbarui API, Anda membuat versi API yang baru. Di API Gateway, Anda memelihara semua versi dokumentasi menggunakan [DocumentationVersions](#) koleksi.

Dokumentasikan API menggunakan konsol API Gateway

Di bagian ini, kami menjelaskan cara membuat dan memelihara bagian dokumentasi API menggunakan konsol API Gateway.

Prasyarat untuk membuat dan mengedit dokumentasi API adalah Anda harus sudah membuat API. Pada bagian ini, kita menggunakan [PetStore](#) API sebagai contoh. Untuk membuat API menggunakan konsol API Gateway, ikuti petunjuk di [Tutorial: Buat REST API dengan mengimpor contoh](#).

Topik

- [Dokumentasikan API entitas](#)
- [Dokumentasikan RESOURCE entitas](#)
- [Dokumentasikan METHOD entitas](#)
- [Dokumentasikan QUERY_PARAMETER entitas](#)
- [Dokumentasikan PATH_PARAMETER entitas](#)
- [Dokumentasikan REQUEST_HEADER entitas](#)
- [Dokumentasikan REQUEST_BODY entitas](#)
- [Dokumentasikan RESPONSE entitas](#)
- [Dokumentasikan RESPONSE_HEADER entitas](#)
- [Dokumentasikan RESPONSE_BODY entitas](#)
- [Dokumentasikan MODEL entitas](#)
- [Dokumentasikan AUTHORIZER entitas](#)

Dokumentasikan **API** entitas

Untuk menambahkan bagian dokumentasi baru untuk API entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih API.

Jika bagian dokumentasi tidak dibuat untuk API, Anda mendapatkan editor properties peta bagian dokumentasi. Masukkan properties peta berikut di editor teks.

```
{
  "info": {
    "description": "Your first API Gateway API.",
    "contact": {
      "name": "John Doe",
      "email": "john.doe@api.com"
    }
  }
}
```

Note

Anda tidak perlu menyandikan properties peta menjadi string JSON. Konsol API Gateway membuat stringifikasi objek JSON untuk Anda.

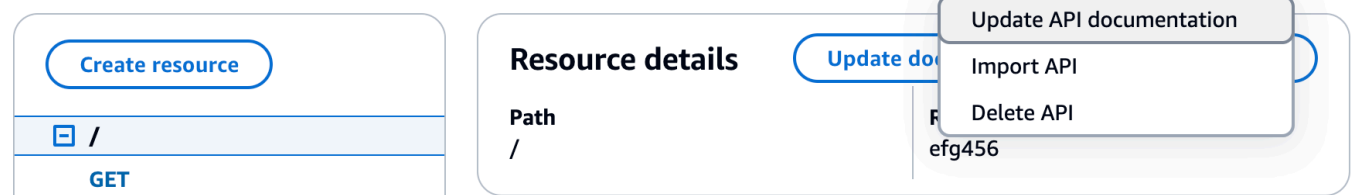
3. Pilih Buat bagian dokumentasi.

Untuk menambahkan bagian dokumentasi baru untuk API entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih menu tindakan API, lalu pilih Perbarui dokumentasi API.

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

Resources



Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Pilih nama API Anda, lalu pada kartu API, pilih Edit.

Dokumentasikan **RESOURCE** entitas

Untuk menambahkan bagian dokumentasi baru untuk RESOURCE entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Sumber Daya.
3. Untuk Path, masukkan jalan.
4. Masukkan deskripsi di editor teks, misalnya:

```
{
  "description": "The PetStore's root resource."
}
```

5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk sumber daya yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk menambahkan bagian dokumentasi baru untuk RESOURCE entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih sumber daya, lalu pilih Perbarui dokumentasi.

Resources

The screenshot shows the 'Resources' panel in the Amazon API Gateway console. On the left, there's a 'Create resource' button and a list of resources, including the root '/' and a resource at '/pets' with a 'GET' method. On the right, the 'Resource details' for the '/pets' resource are shown. It includes a 'Delete' button, an 'Update documentation' button (which is highlighted with a red rectangle), and an 'Enable CORS' button. Below these buttons, the 'Path' is listed as '/pets' and the 'Resource ID' is 'bbb111'. At the top right of the panel, there are 'API actions' and 'Deploy API' buttons.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Pilih sumber daya yang berisi bagian dokumentasi Anda, lalu pilih Edit.

Dokumentasikan **METHOD** entitas

Untuk menambahkan bagian dokumentasi baru untuk METHOD entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Metode.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Masukkan deskripsi di editor teks, misalnya:

```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk metode yang tidak terdaftar.
7. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk menambahkan bagian dokumentasi baru untuk METHOD entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih metode, lalu pilih Perbarui dokumentasi.

[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

[Update documentation](#) [Delete](#)

ARN
arn:aws:execute-api:us-west-2:111222333:abcd1234/*/POST/pets

Resource ID
aaa111

```
graph LR; Client[Client] --> MR[Method request]; MR --> IR[Integration request]; IR --> HTTP[HTTP integration]; HTTP --> IRes[Integration response]; IRes --> MRes[Method response]; MRes --> Client
```

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih metode atau memilih sumber daya yang berisi metode, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **QUERY_PARAMETER** entitas

Untuk menambahkan bagian dokumentasi baru untuk **QUERY_PARAMETER** entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Parameter kueri.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk parameter kueri yang tidak terdaftar.

8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih parameter kueri atau memilih sumber daya yang berisi parameter kueri, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **PATH_PARAMETER** entitas

Untuk menambahkan bagian dokumentasi baru untuk PATH_PARAMETER entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih parameter Path.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk parameter jalur yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih parameter jalur atau memilih sumber daya yang berisi parameter jalur, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **REQUEST_HEADER** entitas

Untuk menambahkan bagian dokumentasi baru untuk REQUEST_HEADER entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Request header.
3. Untuk Path, masukkan jalur untuk header permintaan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk header permintaan yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih header permintaan atau memilih sumber daya yang berisi header permintaan, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **REQUEST_BODY** entitas

Untuk menambahkan bagian dokumentasi baru untuk REQUEST_BODY entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Request body.
3. Untuk Path, masukkan jalur untuk badan permintaan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Masukkan deskripsi di editor teks.
6. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk badan permintaan yang tidak terdaftar.

7. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih badan permintaan atau memilih sumber daya yang berisi badan permintaan, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **RESPONSE** entitas

Untuk menambahkan bagian dokumentasi baru untuk RESPONSE entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Respons (kode status).
3. Untuk Path, masukkan jalur untuk respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk kode status respons yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih kode status respons atau memilih sumber daya yang berisi kode status respons, lalu gunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **RESPONSE_HEADER** entitas

Untuk menambahkan bagian dokumentasi baru untuk **RESPONSE_HEADER** entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih header Response.
3. Untuk Path, masukkan jalur untuk header respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk header respons yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih header respons atau memilih sumber daya yang berisi header respons, lalu gunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **RESPONSE_BODY** entitas

Untuk menambahkan bagian dokumentasi baru untuk **RESPONSE_BODY** entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Badan respons.
3. Untuk Path, masukkan jalur untuk badan respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk badan respons yang tidak terdaftar.

8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih badan respons atau memilih sumber daya yang berisi badan respons, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **MODEL** entitas

Mendokumentasikan MODEL entitas melibatkan pembuatan dan pengelolaan DocumentPart instance untuk model dan masing-masing model '. properties Misalnya, untuk Error model yang disertakan dengan setiap API secara default memiliki definisi skema berikut,

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

dan membutuhkan dua DocumentationPart contoh, satu untuk Model dan yang lainnya untuk message propertinya:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

and

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

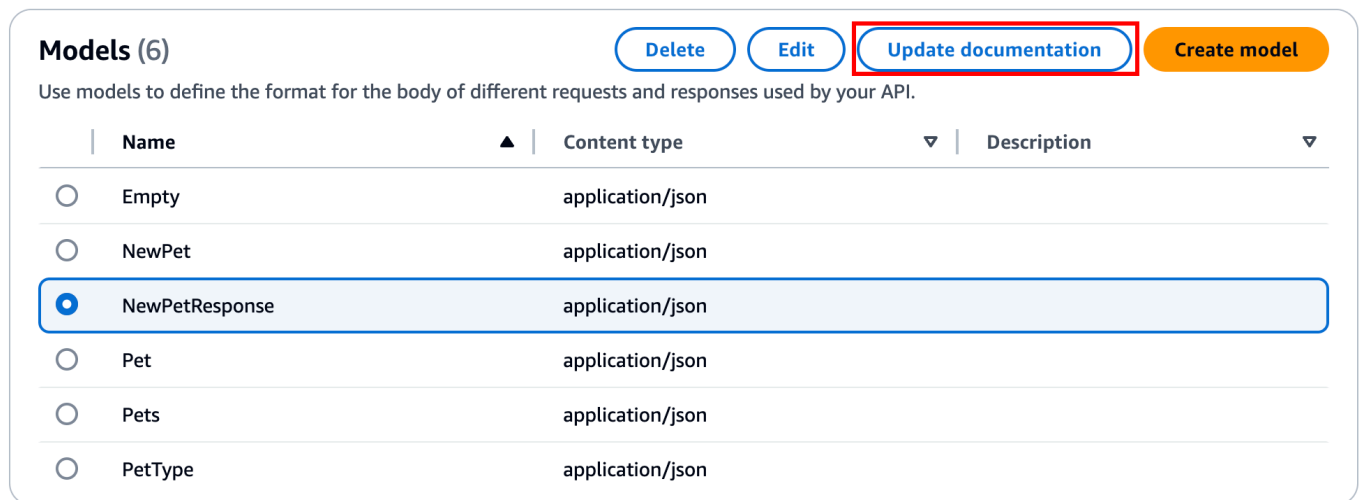
Saat API diekspor, properti akan mengganti nilai dalam skema asli. `DocumentationPart`

Untuk menambahkan bagian dokumentasi baru untuk MODEL entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Model.
3. Untuk Nama, masukkan nama untuk model.
4. Masukkan deskripsi di editor teks.
5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk model yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi ke model lain.

Untuk menambahkan bagian dokumentasi baru untuk MODEL entitas di panel Model, lakukan hal berikut:

1. Di panel navigasi utama, pilih Model.
2. Pilih model, lalu pilih Perbarui dokumentasi.



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	

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Model.
2. Gunakan bilah pencarian atau pilih model, lalu pilih Edit.

Dokumentasikan **AUTHORIZER** entitas

Untuk menambahkan bagian dokumentasi baru untuk AUTHORIZER entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Authorizer.
3. Untuk Nama, masukkan nama otorisasi Anda.
4. Masukkan deskripsi di editor teks. Tentukan nilai untuk `location` bidang yang valid untuk otorisasi.
5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk otorisasi yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi ke otorisasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Authorizers.

- Gunakan bilah pencarian atau pilih otorisasi, lalu pilih Edit.

Publikasikan dokumentasi API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara mempublikasikan versi dokumentasi.

Untuk mempublikasikan versi dokumentasi menggunakan konsol API Gateway

- Di panel navigasi utama, pilih Dokumentasi.
- Pilih Publikasikan dokumentasi.
- Siapkan publikasi:
 - Untuk Stage, pilih panggung.
 - Untuk Versi, masukkan pengenalan versi, misalnya, 1.0.0.
 - (Opsional) Untuk Deskripsi, masukkan deskripsi.
- Pilih Terbitkan.

Anda sekarang dapat melanjutkan untuk mengunduh dokumentasi yang diterbitkan dengan mengekspor dokumentasi ke file OpenAPI eksternal. Untuk mempelajari selengkapnya, lihat [the section called “Ekspor REST API”](#).

Dokumentasikan API menggunakan API Gateway REST API

Di bagian ini, kami menjelaskan cara membuat dan memelihara bagian dokumentasi API menggunakan API Gateway REST API.

Sebelum membuat dan mengedit dokumentasi API, buat dulu API. Pada bagian ini, kita menggunakan [PetStore](#) API sebagai contoh. Untuk membuat API menggunakan konsol API Gateway, ikuti petunjuk di [Tutorial: Buat REST API dengan mengimpor contoh](#).

Topik

- [Dokumentasikan API entitas](#)
- [Dokumentasikan RESOURCE entitas](#)
- [Dokumentasikan METHOD entitas](#)
- [Dokumentasikan QUERY_PARAMETER entitas](#)
- [Dokumentasikan PATH_PARAMETER entitas](#)

- [Dokumentasikan REQUEST_BODY entitas](#)
- [Dokumentasikan REQUEST_HEADER entitas](#)
- [Dokumentasikan RESPONSE entitas](#)
- [Dokumentasikan RESPONSE_HEADER entitas](#)
- [Dokumentasikan AUTHORIZER entitas](#)
- [Dokumentasikan MODEL entitas](#)
- [Perbarui bagian dokumentasi](#)
- [Daftar bagian dokumentasi](#)

Dokumentasikan **API** entitas

Untuk menambahkan dokumentasi untuk [API](#), tambahkan [DocumentationPart](#) resource untuk entitas API:

```
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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  ...
  "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\t}\n\t}"
}
```

Jika bagian dokumentasi telah ditambahkan, 409 Conflict respons kembali, berisi pesan kesalahan. Dalam hal ini, Anda harus memanggil operasi [documentationpart:update](#). Documentation part already exists for the specified location: type 'API'."

```

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\t}\n\t}"
  } ]
}
```

Respons yang berhasil mengembalikan kode 200 OK status dengan payload yang berisi DocumentationPart instance yang diperbarui dalam payload.

Dokumentasikan **RESOURCE** entitas

Untuk menambahkan dokumentasi sumber daya root API, tambahkan sumber [DocumentationPart](#) daya yang ditargetkan untuk sumber [daya Sumber Daya](#) terkait:

```

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",
  },
  "properties" : "{\n\t\"description\" : \"The PetStore root resource.\\n\\n\"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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\\n\"
}
```

Ketika jalur sumber daya tidak ditentukan, sumber daya diasumsikan sebagai sumber daya root. Anda dapat menambahkan "path": "/" properties untuk membuat spesifikasi eksplisit.

Untuk membuat dokumentasi sumber daya turunan API, tambahkan sumber [DocumentationPart](#) yang ditargetkan untuk sumber [daya Sumber Daya](#) terkait:

```
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\n}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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\n}"
}

```

Untuk menambahkan dokumentasi sumber daya turunan yang ditentukan oleh parameter jalur, tambahkan [DocumentationPart](#) sumber daya yang ditargetkan untuk [sumber daya Sumber Daya](#):

```

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/{petId}"
  },
  "properties": "{\n\t\"description\" : \"A child resource specified by the petId
path parameter.\n\n}"
}

```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```

{
  "_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

[DocumentationPart](#) Instance suatu RESOURCE entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Dokumentasikan **METHOD** entitas

Untuk menambahkan dokumentasi untuk metode API, tambahkan [DocumentationPart](#) sumber daya yang ditargetkan untuk sumber daya [Metode](#) yang sesuai:

```

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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:


```
{
  "_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}"
}
```

Jika `location.method` bidang tidak ditentukan dalam permintaan sebelumnya, diasumsikan sebagai ANY metode yang diwakili oleh karakter kartu * liar.

Untuk memperbarui konten dokumentasi METHOD entitas, panggil operasi [documentationpart:update](#), menyediakan peta baru: `properties`

```
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\t\"tags\" : [ \"pets\" ], \n\t\t\"summary\" : \"List all pets.\"\n}"
  } ]
}
```

Respons yang berhasil mengembalikan kode 200 OK status dengan payload yang berisi DocumentationPart instance yang diperbarui dalam payload. Contoh:

```
{
  "_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\t\"tags\" : [ \"pets\" ], \n\t\t\"summary\" : \"List all pets.\"\n}"
}
```

Dokumentasikan **QUERY_PARAMETER** entitas

Untuk menambahkan dokumentasi untuk parameter kueri permintaan, tambahkan [DocumentationPart](#) sumber daya yang ditargetkan untuk QUERY_PARAMETER jenis, dengan bidang yang valid dari path dan name.

```
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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}

```

propertiesPeta bagian dokumentasi QUERY_PARAMETER entitas dapat diwarisi oleh salah satu QUERY_PARAMETER entitas turunannya. Misalnya, jika Anda menambahkan treats sumber daya setelahnya/pets/{petId}, mengaktifkan GET metode/pets/{petId}/treats, dan mengekspos parameter page kueri, parameter kueri anak ini mewarisi properties peta dari parameter kueri yang diberi nama sama dari GET /pets metode tersebut, kecuali jika Anda secara eksplisit menambahkan DocumentationPart sumber daya ke parameter kueri metode tersebutpage. DocumentationPart GET /pets/{petId}/treats

Dokumentasikan **PATH_PARAMETER** entitas

Untuk menambahkan dokumentasi untuk parameter jalur, tambahkan [DocumentationPart](#) sumber daya untuk PATH_PARAMETER entitas.

```

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}"
}

```

```
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}
```

Dokumentasikan **REQUEST_BODY** entitas

Untuk menambahkan dokumentasi untuk badan permintaan, tambahkan [DocumentationPart](#) sumber daya untuk badan permintaan.

```
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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}
```

Dokumentasikan **REQUEST_HEADER** entitas

Untuk menambahkan dokumentasi untuk header permintaan, tambahkan [DocumentationPart](#) sumber daya untuk header permintaan.

```
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}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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\nmethod invocation.\"\n}"
}

```

Dokumentasikan **RESPONSE** entitas

Untuk menambahkan dokumentasi untuk respons kode status, tambahkan

[DocumentationPart](#) sumber daya yang ditargetkan untuk sumber [MethodResponse](#) daya yang sesuai.

```

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}"
}

```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:


```
{
  "_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}"
}
```

Dokumentasikan **RESPONSE_HEADER** entitas

Untuk menambahkan dokumentasi untuk header respons, tambahkan [DocumentationPart](#) sumber daya untuk header respons.

```
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}"
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}
```

Dokumentasi header Content-Type respons ini adalah dokumentasi default untuk Content-Type header dari setiap respons API.

Dokumentasikan **AUTHORIZER** entitas

Untuk menambahkan dokumentasi bagi otorisasi API, tambahkan [DocumentationPart](#) sumber daya yang ditargetkan untuk otorisasi yang ditentukan.

```
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\n}"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi DocumentationPart instance yang baru dibuat di payload. Contoh:

```
{
  "_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\t\n}"
}

```

Note

[DocumentationPart](#) Instance suatu AUTHORIZER entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Dokumentasikan **MODEL** entitas

Mendokumentasikan MODEL entitas melibatkan pembuatan dan pengelolaan DocumentPart instance untuk model dan masing-masing model ' properties Misalnya, untuk Error model yang disertakan dengan setiap API secara default memiliki definisi skema berikut,

```

{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}

```

dan membutuhkan dua DocumentationPart contoh, satu untuk Model dan yang lainnya untuk message propertinya:

```

{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}

```

and

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Saat API diekspor, properti akan mengganti nilai dalam skema asli. `DocumentationPart`

Untuk menambahkan dokumentasi untuk model API, tambahkan [DocumentationPart](#) sumber daya yang ditargetkan untuk model yang ditentukan.

```
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" : "MODEL",
    "name" : "Pet"
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\\n\\n\"
}
```

Jika berhasil, operasi mengembalikan 201 Created respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Contoh:

```
{
  "_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}"
}

```

Ulangi langkah yang sama untuk membuat DocumentationPart instance untuk salah satu properti model.

Note

[DocumentationPart](#) instance suatu MODEL entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Perbarui bagian dokumentasi

Untuk memperbarui bagian dokumentasi dari semua jenis entitas API, kirimkan permintaan PATCH pada [DocumentationPart](#) instance pengidentifikasi bagian tertentu untuk mengganti properties peta yang ada dengan yang baru.

```

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" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}
```

Respons yang berhasil mengembalikan kode 200 OK status dengan payload yang berisi DocumentationPart instance yang diperbarui dalam payload.

Anda dapat memperbarui beberapa bagian dokumentasi dalam satu PATCH permintaan.

Daftar bagian dokumentasi

Untuk mencantumkan bagian dokumentasi dari semua jenis entitas API, kirimkan permintaan GET pada [DocumentationParts](#) koleksi.

```
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
```

Respons yang berhasil mengembalikan kode 200 OK status dengan payload yang berisi DocumentationPart instance yang tersedia di payload.

Publikasikan dokumentasi API menggunakan API Gateway REST API

Untuk memublikasikan dokumentasi untuk API, buat, perbarui, atau dapatkan snapshot dokumentasi, lalu kaitkan snapshot dokumentasi dengan tahap API. Saat membuat snapshot dokumentasi, Anda juga dapat mengaitkannya dengan tahap API secara bersamaan.

Topik

- [Buat snapshot dokumentasi dan kaitkan dengan tahap API](#)

- [Buat snapshot dokumentasi](#)
- [Perbarui snapshot dokumentasi](#)
- [Dapatkan snapshot dokumentasi](#)
- [Kaitkan snapshot dokumentasi dengan tahap API](#)
- [Unduh snapshot dokumentasi yang terkait dengan panggung](#)

Buat snapshot dokumentasi dan kaitkan dengan tahap API

Untuk membuat snapshot bagian dokumentasi API dan mengaitkannya dengan tahap API secara bersamaan, kirimkan POST permintaan berikut:

```
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"
}
```

Jika berhasil, operasi mengembalikan 200 OK respons, yang berisi DocumentationVersion instance yang baru dibuat sebagai muatan.

Atau, Anda dapat membuat snapshot dokumentasi tanpa mengaitkannya dengan tahap API terlebih dahulu dan kemudian memanggil [restapi:update](#) untuk mengaitkan snapshot dengan tahap API tertentu. Anda juga dapat memperbarui atau menanyakan snapshot dokumentasi yang ada dan kemudian memperbarui asosiasi tahapannya. Kami menunjukkan langkah-langkah di empat bagian berikutnya.

Buat snapshot dokumentasi

Untuk membuat snapshot bagian dokumentasi API, buat [DocumentationVersion](#) resource baru dan tambahkan ke [DocumentationVersions](#) koleksi API:


```

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"
}

```

Jika berhasil, operasi mengembalikan 200 OK respons, yang berisi DocumentationVersion instance yang baru dibuat sebagai muatan.

Perbarui snapshot dokumentasi

Anda hanya dapat memperbarui snapshot dokumentasi dengan memodifikasi description properti sumber daya yang sesuai [DocumentationVersion](#). Contoh berikut menunjukkan cara memperbarui deskripsi snapshot dokumentasi seperti yang diidentifikasi oleh pengenal versinya, misalnya *version*, 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."
  }]
}

```

Jika berhasil, operasi mengembalikan 200 OK respons, yang berisi DocumentationVersion instance yang diperbarui sebagai muatan.

Dapatkan snapshot dokumentasi

Untuk mendapatkan snapshot dokumentasi, kirimkan GET permintaan terhadap [DocumentationVersion](#) sumber daya yang ditentukan. Contoh berikut menunjukkan cara mendapatkan snapshot dokumentasi dari pengenalan versi tertentu, 1.0.0.

```
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
```

Kaitkan snapshot dokumentasi dengan tahap API

Untuk mempublikasikan dokumentasi API, kaitkan snapshot dokumentasi dengan tahap API. Anda harus sudah membuat tahap API sebelum mengaitkan versi dokumentasi dengan stage.

Untuk mengaitkan snapshot dokumentasi dengan tahap API menggunakan API [Gateway REST API](#), panggil operasi [stage:update](#) untuk menyetel versi dokumentasi yang diinginkan di properti: `stage.documentationVersion`

```
PATCH /restapis/RESTAPI_ID/stages/STAGE_NAME
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": "/documentationVersion",
    "value": "VERSION_IDENTIFIER"
  }]
}
```

Unduh snapshot dokumentasi yang terkait dengan panggung

Setelah versi bagian dokumentasi dikaitkan dengan tahapan, Anda dapat mengekspor bagian dokumentasi bersama dengan definisi entitas API, ke file eksternal, menggunakan konsol API Gateway, API Gateway REST API, salah satunya SDKs, atau AWS CLI untuk API Gateway. Prosesnya sama dengan mengekspor API. Format file yang diekspor dapat berupa JSON atau YAMB.

Menggunakan API Gateway REST API, Anda juga dapat secara eksplisit menyetel parameter `extension=documentation,integrations,authorizers` kueri untuk menyertakan bagian dokumentasi API, integrasi API, dan otorisasi dalam ekspor API. Secara default, bagian dokumentasi disertakan, tetapi integrasi dan otorisasi dikecualikan, saat Anda mengekspor API. Output default dari ekspor API cocok untuk distribusi dokumentasi.

Untuk mengekspor dokumentasi API dalam file OpenAPI JSON eksternal menggunakan API Gateway REST API, kirimkan permintaan berikut GET:

```
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: 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
```

Di sini, `x-amazon-apigateway-documentation` objek berisi bagian dokumentasi dan definisi entitas API berisi properti dokumentasi yang didukung oleh OpenAPI. Output tidak termasuk rincian integrasi atau otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus). Untuk memasukkan kedua detail, atur `extensions=integrations,authorizers,documentation`. Untuk menyertakan detail integrasi tetapi bukan otorisasi, atur `extensions=integrations,documentation`.

Anda harus mengatur `Accept:application/json` header dalam permintaan untuk menampilkan hasil dalam file JSON. Untuk menghasilkan output YAMM, ubah header permintaan menjadi `Accept:application/yaml`.

Sebagai contoh, kita akan melihat API yang mengekspos GET metode sederhana pada sumber daya root (/). API ini memiliki empat entitas API yang didefinisikan dalam file definisi OpenAPI, satu untuk

masing-masing API, METHOD, dan RESPONSE tipe. Bagian dokumentasi telah ditambahkan ke masing-masing API, METHOD, dan RESPONSE entitas. Memanggil perintah `documentation-exporting` sebelumnya, kita mendapatkan output berikut, dengan bagian-bagian dokumentasi yang tercantum dalam `x-amazon-apigateway-documentation` objek sebagai ekstensi ke file OpenAPI standar.

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://rznaap68yi.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"
}

```

Untuk atribut yang sesuai dengan OpenAPI yang ditentukan dalam properties peta bagian dokumentasi, API Gateway menyisipkan atribut ke dalam definisi entitas API terkait. Atribut *x-something* adalah ekstensi OpenAPI standar. Ekstensi ini disebarkan ke dalam definisi entitas API. Misalnya, lihat *x-example* atribut untuk GET metode. Atribut seperti *foo* bukan bagian dari spesifikasi OpenAPI dan tidak disuntikkan ke dalam definisi entitas API terkait.

Jika alat rendering dokumentasi (misalnya, [OpenAPI UI](#)) [mem-parsing definisi entitas API](#) untuk mengekstrak atribut dokumentasi, atribut 'instance' yang tidak sesuai dengan OpenAPI properties tidak tersedia untuk alat tersebut. DocumentationPart Namun, jika alat rendering dokumentasi mem-parsing *x-amazon-apigateway-documentation* objek untuk mendapatkan konten, atau jika alat memanggil [restapi:documentation-parts](#) dan [documentationpart:by-id](#) untuk [mengambil bagian dokumentasi dari API Gateway, semua atribut dokumentasi tersedia](#) untuk ditampilkan oleh alat tersebut.

Untuk mengekspor dokumentasi dengan definisi entitas API yang berisi detail integrasi ke file OpenAPI JSON, kirimkan permintaan berikut GET:

```
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
```

Untuk mengekspor dokumentasi dengan definisi entitas API yang berisi detail integrasi dan otorisasi ke file OpenAPI YAMAL, kirimkan permintaan berikut: GET

```
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
```

Untuk menggunakan konsol API Gateway untuk mengekspor dan mengunduh dokumentasi API yang dipublikasikan, ikuti petunjuk di [Ekspor REST API menggunakan konsol API Gateway](#).

Impor dokumentasi API

Seperti halnya mengimpor definisi entitas API, Anda dapat mengimpor bagian dokumentasi dari file OpenAPI eksternal ke API di API Gateway. Anda menentukan bagian to-be-imported dokumentasi dalam [x-amazon-apigateway-documentation objek](#) ekstensi dalam file definisi OpenAPI yang valid. Mengimpor dokumentasi tidak mengubah definisi entitas API yang ada.

Anda memiliki opsi untuk menggabungkan bagian dokumentasi yang baru ditentukan ke dalam bagian dokumentasi yang ada di API Gateway atau menimpa bagian dokumentasi yang ada. Dalam MERGE mode, bagian dokumentasi baru yang ditentukan dalam file OpenAPI ditambahkan ke DocumentationParts koleksi API. Jika impor DocumentationPart sudah ada, atribut yang

diimpor menggantikan yang sudah ada jika keduanya berbeda. Atribut dokumentasi lain yang ada tetap tidak terpengaruh. Dalam OVERWRITE mode, seluruh DocumentationParts koleksi diganti sesuai dengan file definisi OpenAPI yang diimpor.

Mengimpor bagian dokumentasi menggunakan API Gateway REST API

Untuk mengimpor dokumentasi API menggunakan API Gateway REST API, panggil operasi [documentationpart:import](#). Contoh berikut menunjukkan cara menimpa bagian dokumentasi API yang ada dengan satu GET / metode, mengembalikan 200 OK respons saat berhasil.

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"
    }
  }
]
}
}

```

Jika berhasil, permintaan ini mengembalikan respons 200 OK yang berisi impor `DocumentationPartId` dalam muatan.

```

{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}

```

Selain itu, Anda juga dapat memanggil [restapi:import](#) atau [restapi:put](#), memasok bagian dokumentasi dalam objek `x-amazon-apigateway-documentation` sebagai bagian dari file OpenAPI input dari definisi API. Untuk mengecualikan bagian dokumentasi dari impor API, atur `ignore=documentation` parameter kueri permintaan.

Mengimpor bagian dokumentasi menggunakan konsol API Gateway

Petunjuk berikut menjelaskan cara mengimpor bagian dokumentasi.

Untuk menggunakan konsol untuk mengimpor bagian dokumentasi API dari file eksternal

1. Di panel navigasi utama, pilih Dokumentasi.
2. Pilih Impor.
3. Jika Anda memiliki dokumentasi yang sudah ada, pilih untuk Menimpa atau Menggabungkan dokumentasi baru Anda.
4. Pilih file untuk memuat file dari drive, atau masukkan konten file ke dalam tampilan file. Sebagai contoh, lihat payload permintaan contoh di [Mengimpor bagian dokumentasi menggunakan API Gateway REST API](#).
5. Pilih cara menangani peringatan saat impor. Pilih salah satu Gagal pada peringatan atau Abaikan peringatan. Untuk informasi selengkapnya, lihat [the section called “Kesalahan dan peringatan dari mengimpor API Anda ke API Gateway”](#).
6. Pilih Impor.

Kontrol akses ke dokumentasi API di API Gateway

Jika Anda memiliki tim dokumentasi khusus untuk menulis dan mengedit dokumentasi API Anda, Anda dapat mengonfigurasi izin akses terpisah untuk pengembang Anda (untuk pengembangan API) dan untuk penulis atau editor Anda (untuk pengembangan konten). Ini sangat tepat ketika vendor pihak ketiga terlibat dalam membuat dokumentasi untuk Anda.

Untuk memberi tim dokumentasi Anda akses untuk membuat, memperbarui, dan memublikasikan dokumentasi API Anda, Anda dapat menetapkan peran IAM kepada tim dokumentasi dengan kebijakan IAM berikut, di mana *account_id* ID AWS akun tim dokumentasi Anda.

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/*"
]
}
]
```

Untuk informasi tentang menyetel izin untuk mengakses sumber daya API Gateway, lihat [the section called “Cara kerja Amazon API Gateway dengan IAM”](#).

Hasilkan SDKs untuk REST APIs di API Gateway

Untuk memanggil REST API Anda dengan cara khusus platform atau bahasa, Anda harus membuat SDK API khusus platform atau bahasa. Anda menghasilkan SDK setelah membuat, menguji, dan menerapkan API ke suatu panggung. Saat ini, API Gateway mendukung pembuatan SDK untuk API di Java, Java untuk Android JavaScript, Objective-C atau Swift untuk iOS, dan Ruby.

Bagian ini menjelaskan cara membuat SDK API Gateway API. Ini juga menunjukkan cara menggunakan SDK yang dihasilkan di aplikasi Java, aplikasi Java untuk Android, Objective-C dan Swift untuk aplikasi iOS, dan aplikasi. JavaScript

Untuk memfasilitasi diskusi, kami menggunakan API Gateway [API](#) ini, yang memperlihatkan fungsi Lambda [Kalkulator Sederhana](#) ini.

Sebelum melanjutkan, buat atau impor API dan terapkan setidaknya sekali di API Gateway. Untuk petunjuk, lihat [Menerapkan REST APIs di API Gateway](#).

Topik

- [Kalkulator sederhana fungsi Lambda](#)
- [API kalkulator sederhana di API Gateway](#)
- [Definisi OpenAPI API kalkulator sederhana](#)
- [Menghasilkan Java SDK API di API Gateway](#)

- [Menghasilkan SDK Android API di API Gateway](#)
- [Menghasilkan SDK iOS API di API Gateway](#)
- [Menghasilkan JavaScript SDK REST API di API Gateway](#)
- [Menghasilkan Ruby SDK API di API Gateway](#)
- [SDKs Menghasilkan API menggunakan AWS CLI perintah di API Gateway](#)

Kalkulator sederhana fungsi Lambda

Sebagai ilustrasi, kita akan menggunakan fungsi Lambda Node.js yang melakukan operasi biner penambahan, pengurangan, perkalian dan pembagian.

Topik

- [Kalkulator sederhana Format input fungsi Lambda](#)
- [Kalkulator sederhana Format output fungsi Lambda](#)
- [Implementasi fungsi Lambda kalkulator sederhana](#)

Kalkulator sederhana Format input fungsi Lambda

Fungsi ini mengambil input dari format berikut:

```
{ "a": "Number", "b": "Number", "op": "string" }
```

di mana op bisa menjadi salah satu dari(+, -, *, /, add, sub, mul, div).

Kalkulator sederhana Format output fungsi Lambda

Ketika operasi berhasil, ia mengembalikan hasil dari format berikut:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number" }
```

dimana c memegang hasil perhitungan.

Implementasi fungsi Lambda kalkulator sederhana

Implementasi fungsi Lambda adalah sebagai berikut:

```
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";
}

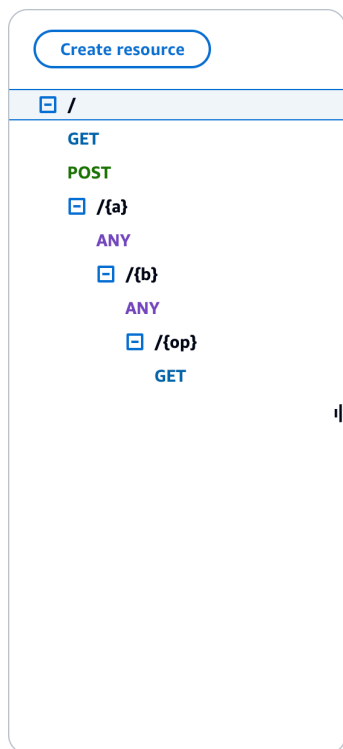
const res = {};
res.a = Number(event.a);
res.b = Number(event.b);
res.op = event.op;
if (isNaN(event.a) || isNaN(event.b)) {
  return "400 Invalid Operand";
}
switch (event.op) {
  case "+":
  case "add":
    res.c = res.a + res.b;
    break;
  case "-":
  case "sub":
    res.c = res.a - res.b;
    break;
  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;
};
```

API kalkulator sederhana di API Gateway

API kalkulator sederhana kami mengekspos tiga metode (GET, POST, GET) untuk memanggil. [the section called “Kalkulator sederhana fungsi Lambda”](#) Representasi grafis dari API ini ditampilkan sebagai berikut:

Resources



Ketiga metode ini menunjukkan cara berbeda untuk memasok input untuk fungsi Lambda backend untuk melakukan operasi yang sama:

- GET `/?a=...&b=...&op=...` Metode ini menggunakan parameter kueri untuk menentukan input.
- POST `/` Metode ini menggunakan payload JSON `{"a": "Number", "b": "Number", "op": "string"}` untuk menentukan input.
- GET `/a/b/op` Metode ini menggunakan parameter jalur untuk menentukan input.

Jika tidak ditentukan, API Gateway menghasilkan nama metode SDK yang sesuai dengan menggabungkan metode HTTP dan bagian jalur. Bagian jalur root (`/`) disebut sebagai `Api Root`. Misalnya, nama metode Java SDK default untuk metode API GET `/?a=...&b=...&op=...` adalah `getABOp`, nama metode SDK default untuk POST `/` adalah `postApiRoot`, dan nama metode

SDK default untuk adalah. `GET /{a}/{b}/{op} getABOp` Individu SDKs dapat menyesuaikan konvensi. Lihat dokumentasi di sumber SDK yang dihasilkan untuk nama metode khusus SDK.

Anda dapat, dan harus, mengganti nama metode SDK default dengan menentukan properti `operationName` pada setiap metode [API](#). Anda dapat melakukannya saat [membuat metode API atau memperbarui metode API](#) menggunakan API Gateway REST API. Dalam definisi API Swagger, Anda dapat mengatur `operationId` untuk mencapai hasil yang sama.

Sebelum menunjukkan cara memanggil metode ini menggunakan SDK yang dihasilkan oleh API Gateway untuk API ini, mari kita ingat secara singkat cara mengaturnya. Untuk petunjuk mendetail, lihat [Kembangkan REST API di API Gateway](#). Jika Anda baru mengenal API Gateway, lihat [Pilih tutorial AWS Lambda integrasi](#) terlebih dahulu.

Buat model untuk input dan output

Untuk menentukan input yang diketik dengan kuat di SDK, kami membuat Input model untuk API. Untuk menggambarkan tipe data badan respons, kami membuat Output model dan Result model.

Untuk membuat model untuk input, output, dan hasil

1. Di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **input**.
4. Untuk jenis Konten, masukkan **application/json**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan **default**.

5. Untuk skema Model, masukkan model berikut:

```
{
  "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "a": { "type": "number" },
    "b": { "type": "number" },
    "op": { "type": "string" }
  },
  "title": "Input"
}
```

6. Pilih Buat model.
7. Ulangi langkah-langkah berikut untuk membuat Output model dan Result model.

Untuk Output model, masukkan yang berikut ini untuk skema Model:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "c": {"type": "number"}
  },
  "title": "Output"
}
```

Untuk Result model, masukkan yang berikut ini untuk skema Model. Ganti ID API abc123 dengan ID API Anda.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "input": {
      "$ref": "https://apigateway.amazonaws.com/restapis/abc123/models/Input"
    },
    "output": {
      "$ref": "https://apigateway.amazonaws.com/restapis/abc123/models/Output"
    }
  },
  "title": "Result"
}
```

Siapkan GET/parameter kueri metode

Untuk GET /?a=..&b=..&op=.. metode ini, parameter kueri dideklarasikan dalam Permintaan Metode:

Untuk mengatur parameter string kueri GET/URL

1. Di bagian Permintaan metode untuk GET metode pada sumber daya root (/), pilih Edit.
2. Pilih parameter string kueri URL dan lakukan hal berikut:

- a. Pilih Tambahkan string kueri.
- b. Untuk Nama, masukkan **a**.
- c. Tetap Diperlukan dan Caching dimatikan.
- d. Tetap caching dimatikan.

Ulangi langkah yang sama dan buat string kueri bernama **b** dan string kueri bernama **op**.

3. Pilih Simpan.

Siapkan model data untuk payload sebagai masukan ke backend

Untuk POST / metode ini, kita membuat Input model dan menambahkannya ke permintaan metode untuk menentukan bentuk data input.

Untuk mengatur model data untuk payload sebagai masukan ke backend

1. Di bagian Permintaan metode, untuk POST metode pada sumber daya root (/) pilih Edit.
2. Pilih badan Permintaan.
3. Pilih Tambah model.
4. Untuk jenis Konten, masukkan **application/json**.
5. Untuk Model, pilih Input.
6. Pilih Simpan.

Dengan model ini, pelanggan API Anda dapat memanggil SDK untuk menentukan input dengan membuat instance objek. Input Tanpa model ini, pelanggan Anda akan diminta untuk membuat objek kamus untuk mewakili input JSON ke fungsi Lambda.

Mengatur model data untuk output hasil dari backend

Untuk ketiga metode, kita membuat Result model dan menambahkannya ke metode Method Response untuk menentukan bentuk output yang dikembalikan oleh fungsi Lambda.

Untuk mengatur model data untuk output hasil dari backend

1. Pilih sumber daya `{a}/{b}/{op}`, lalu pilih metode GET.
2. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.

3. Di bawah Badan respons, pilih Tambahkan model.
4. Untuk jenis Konten, masukkan **application/json**.
5. Untuk Model, pilih Hasil.
6. Pilih Simpan.

Dengan model ini, pelanggan API Anda dapat mengurai output yang berhasil dengan membaca properti `Result` objek. Tanpa model ini, pelanggan akan diminta untuk membuat objek kamus untuk mewakili output JSON.

Definisi OpenAPI API kalkulator sederhana

Berikut ini adalah definisi OpenAPI dari API kalkulator sederhana. Anda dapat mengimpornya ke akun Anda. Namun, Anda perlu mengatur ulang izin berbasis sumber daya pada fungsi [Lambda](#) setelah impor. Untuk melakukannya, pilih kembali fungsi Lambda yang Anda buat di akun dari Permintaan Integrasi di konsol API Gateway. Ini akan menyebabkan konsol API Gateway mengatur ulang izin yang diperlukan. [Atau, Anda dapat menggunakan AWS Command Line Interface untuk Lambda perintah `add-permission`.](#)

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\" : $input.params('b'),\n  \"op\" :\n    \"$input.params('op')\"\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_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
    \"input\" : {\n        \"a\" : $inputRoot.a,\n        \"b\" : $inputRoot.b,\n        \"op\" :
    \"$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
\\\"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"
    }
  },
  {
    "/": {
      "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
\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"
}
},
"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\" : $inputRoot.b,\n    \"op\" :
\n\"$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"
      }
    }
  }
}
```

```
}  
}  
}  
}  
}  
}
```

Menghasilkan Java SDK API di API Gateway

Prosedur berikut menunjukkan cara membuat Java SDK API di API Gateway.

Untuk menghasilkan Java SDK API di API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Java dan lakukan hal berikut:
 - a. Untuk Nama Layanan, tentukan nama SDK Anda. Misalnya, **SimpleCalcSdk**. Ini menjadi nama kelas klien SDK Anda. Nama sesuai dengan `<name>` tag di bawah `<project>` dalam file `pom.xml`, yang ada di folder proyek SDK. Jangan sertakan tanda hubung.
 - b. Untuk Java Package Name, tentukan nama paket untuk SDK Anda. Misalnya, **examples.aws.apig.simpleCalc.sdk**. Nama paket ini digunakan sebagai namespace pustaka SDK Anda. Jangan sertakan tanda hubung.
 - c. Untuk Java Build System, masukkan **maven** atau **gradle** tentukan sistem build.
 - d. Untuk Id Grup Java, masukkan pengenalan grup untuk proyek SDK Anda. Misalnya, masukkan **my-apig-api-examples**. Pengidentifikasi ini sesuai dengan `<groupId>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.
 - e. Untuk Id Artifact Java, masukkan pengenalan artefak untuk proyek SDK Anda. Misalnya, masukkan **simple-calc-sdk**. Pengidentifikasi ini sesuai dengan `<artifactId>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.
 - f. Untuk Versi Artifact Java, masukkan string pengenalan versi. Misalnya, **1.0.0**. Pengidentifikasi versi ini sesuai dengan `<version>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.

- g. Untuk Teks Lisensi Kode Sumber, masukkan teks lisensi kode sumber Anda, jika ada.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk [Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API](#) untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan SDK Android API di API Gateway

Prosedur berikut menunjukkan cara membuat Android SDK API di API Gateway.

Untuk menghasilkan SDK Android API di API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Android dan lakukan hal berikut:
 - a. Untuk ID Grup, masukkan pengenal unik untuk proyek terkait. Ini digunakan dalam `pom.xml` file (misalnya, **com.mycompany**).
 - b. Untuk paket Invoker, masukkan namespace untuk kelas klien yang dihasilkan (misalnya, **com.mycompany.clientsdk**).
 - c. Untuk Artifact ID, masukkan nama file.jar yang dikompilasi tanpa versi. Ini digunakan dalam `pom.xml` file (misalnya, **aws-apigateway-api-sdk**).
 - d. Untuk versi Artifact, masukkan nomor versi artefak untuk klien yang dihasilkan. Ini digunakan dalam `pom.xml` file dan harus mengikuti *major.minor.patch* pola (misalnya, **1.0.0**).
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk [Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API](#) untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan SDK iOS API di API Gateway

Prosedur berikut menunjukkan cara membuat SDK iOS dari API di API Gateway.

Untuk menghasilkan SDK iOS API di API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform iOS (Objective-C) atau iOS (Swift) dan lakukan hal berikut:
 - Ketik awalan unik di kotak Awalan.

Efek awalan adalah sebagai berikut: jika Anda menetapkan, misalnya, **SIMPLE_CALC** sebagai awalan untuk SDK [SimpleCalcAPI](#) dengan, dan result model inputoutput, SDK yang dihasilkan akan berisi SIMPLE_CALCSimpleCalcClient kelas yang merangkum API, termasuk permintaan/tanggapan metode. Selain itu, SDK yang dihasilkan akan berisiSIMPLE_CALCinput,SIMPLE_CALCoutput, dan SIMPLE_CALCresult kelas untuk mewakili input, output, dan hasil, masing-masing, untuk mewakili input permintaan dan output respons. Untuk informasi selengkapnya, lihat [Menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift](#).

7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk [Menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift](#) untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan JavaScript SDK REST API di API Gateway

Prosedur berikut menunjukkan cara membuat JavaScript SDK API di API Gateway.

Untuk menghasilkan JavaScript SDK API di API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih JavaScriptplatformnya.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk [Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API](#) untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan Ruby SDK API di API Gateway

Prosedur berikut menunjukkan cara membuat Ruby SDK API di API Gateway.

Untuk menghasilkan Ruby SDK API di API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Ruby dan lakukan hal berikut:
 - a. Untuk Nama Layanan, tentukan nama SDK Anda. Misalnya, **SimpleCalc**. Ini digunakan untuk menghasilkan namespace Ruby Gem dari API Anda. Nama harus semua huruf, (a-zA-Z), tanpa karakter atau angka khusus lainnya.

- b. Untuk Nama Permata Ruby, tentukan nama Ruby Gem yang berisi kode sumber SDK yang dihasilkan untuk API Anda. Secara default, ini adalah nama layanan dengan cache yang lebih rendah ditambah `-sdk` sufiks—misalnya, **simplecalc-sdk**
 - c. Untuk Ruby Gem Version, tentukan nomor versi untuk Ruby Gem yang dihasilkan. Secara default, nilainya diatur ke `1.0.0`.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk [Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API](#) untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

SDKs Menghasilkan API menggunakan AWS CLI perintah di API Gateway

Anda dapat menggunakan AWS CLI untuk membuat dan mengunduh SDK API untuk platform yang didukung dengan memanggil perintah [get-sdk](#). Kami mendemonstrasikan ini untuk beberapa platform yang didukung berikut ini.

Topik

- [Buat dan unduh Java untuk Android SDK menggunakan AWS CLI](#)
- [Buat dan unduh JavaScript SDK menggunakan AWS CLI](#)
- [Buat dan unduh Ruby SDK menggunakan AWS CLI](#)

Buat dan unduh Java untuk Android SDK menggunakan AWS CLI

Untuk membuat dan mengunduh Java untuk Android SDK yang dihasilkan oleh API Gateway dari API (udpuvvzbkc) pada tahap tertentu (test), panggil perintah sebagai berikut:

```
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
```

Input terakhir dari `~/apps/myApi/myApi-android-sdk.zip` adalah path ke file SDK yang diunduh bernamamyApi-android-sdk.zip.

Buat dan unduh JavaScript SDK menggunakan AWS CLI

Untuk membuat dan mengunduh JavaScript SDK yang dihasilkan oleh API Gateway dari API (udpuvvzbkc) pada tahap tertentu (test), panggil perintah sebagai berikut:

```
aws apigateway get-sdk \
    --rest-api-id udpuvvzbkc \
    --stage-name test \
    --sdk-type javascript \
    ~/apps/myApi/myApi-js-sdk.zip
```

Input terakhir dari `~/apps/myApi/myApi-js-sdk.zip` adalah path ke file SDK yang diunduh bernamamyApi-js-sdk.zip.

Buat dan unduh Ruby SDK menggunakan AWS CLI

Untuk membuat dan mengunduh Ruby SDK dari API (udpuvvzbkc) pada tahap tertentu (test), panggil perintah sebagai berikut:

```
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
```

Input terakhir dari `~/apps/myApi/myApi-ruby-sdk.zip` adalah path ke file SDK yang diunduh bernamamyApi-ruby-sdk.zip.

Selanjutnya, kami menunjukkan cara menggunakan SDK yang dihasilkan untuk memanggil API yang mendasarinya. Lihat informasi yang lebih lengkap di [Panggil REST APIs di API Gateway](#).

Jual API Gateway Anda APIs melalui AWS Marketplace

Setelah Anda membangun, menguji, dan menerapkan APIs, Anda dapat mengemasnya dalam paket [penggunaan](#) API Gateway dan menjual paket tersebut sebagai produk Perangkat Lunak sebagai

Layanan (SaaS). AWS Marketplace Pembeli API yang berlangganan penawaran produk Anda ditagih AWS Marketplace berdasarkan jumlah permintaan yang dibuat untuk paket penggunaan.

Untuk menjual APIs on Anda AWS Marketplace, Anda harus menyiapkan saluran penjualan untuk diintegrasikan AWS Marketplace dengan API Gateway. Secara umum, ini melibatkan daftar produk Anda AWS Marketplace, menyiapkan peran IAM dengan kebijakan yang sesuai untuk memungkinkan API Gateway mengirim metrik penggunaan AWS Marketplace, mengaitkan AWS Marketplace produk dengan paket penggunaan API Gateway, dan mengaitkan AWS Marketplace pembeli dengan kunci API Gateway API. Rincian dibahas di bagian berikut.

Untuk informasi selengkapnya tentang menjual API Anda sebagai produk SaaS AWS Marketplace, lihat [AWS Marketplace Panduan Pengguna](#).

Topik

- [Inisialisasi AWS Marketplace integrasi dengan API Gateway](#)
- [Menangani langganan pelanggan ke paket penggunaan](#)

Inisialisasi AWS Marketplace integrasi dengan API Gateway

Tugas berikut adalah untuk inisialisasi AWS Marketplace integrasi satu kali dengan API Gateway, yang memungkinkan Anda menjual APIs sebagai produk SaaS.

Daftar produk di AWS Marketplace

Untuk mencantumkan paket penggunaan Anda sebagai produk SaaS, kirimkan formulir pemuatan produk melalui [AWS Marketplace](#) Produk harus berisi dimensi yang diberi nama `apigateway` dari `requests` jenisnya. Dimensi ini mendefinisikan `price-per-request` dan digunakan oleh API Gateway untuk meteran permintaan ke Anda APIs.

Buat peran pengukuran

Buat peran IAM bernama `ApiGatewayMarketplaceMeteringRole` dengan kebijakan eksekusi dan kebijakan kepercayaan berikut. Peran ini memungkinkan API Gateway mengirim metrik penggunaan atas nama Anda. AWS Marketplace

Kebijakan eksekusi peran pengukuran

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "aws-marketplace:BatchMeterUsage",
      "aws-marketplace:ResolveCustomer"
    ],
    "Resource": "*",
    "Effect": "Allow"
  }
]
}

```

Kebijakan hubungan terpercaya dari peran pengukuran

JSON

```

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

```

Mengaitkan rencana penggunaan dengan AWS Marketplace produk

Saat Anda mencantumkan produk AWS Marketplace, Anda menerima kode AWS Marketplace produk. Untuk mengintegrasikan API Gateway dengan AWS Marketplace, kaitkan paket penggunaan Anda dengan kode AWS Marketplace produk. Anda mengaktifkan asosiasi dengan menyetel [productCode](#) bidang API UsagePlan Gateway ke kode AWS Marketplace produk Anda, menggunakan konsol API Gateway, API Gateway REST API, API Gateway AWS CLI for, atau AWS SDK for API Gateway. Contoh kode berikut menggunakan API Gateway REST API:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "patchOperations" : [{
    "path" : "/productCode",
    "value" : "MARKETPLACE_PRODUCT_CODE",
    "op" : "replace"
  }]
}
```

Menangani langganan pelanggan ke paket penggunaan

Tugas-tugas berikut ditangani oleh aplikasi portal pengembang Anda.

Saat pelanggan berlangganan produk Anda AWS Marketplace, AWS Marketplace teruskan POST permintaan ke URL langganan SaaS yang Anda daftarkan saat mencantumkan produk Anda. AWS Marketplace POST Permintaan dilengkapi dengan `x-amzn-marketplace-token` parameter yang berisi informasi pembeli. Ikuti petunjuk dalam [orientasi pelanggan SaaS untuk menangani pengalihan](#) ini di aplikasi portal pengembang Anda.

Menanggapi permintaan berlangganan pelanggan, AWS Marketplace mengirimkan `subscribe-success` pemberitahuan ke topik Amazon SNS yang dapat Anda berlangganan. (Lihat [orientasi pelanggan SaaS](#)). Untuk menerima permintaan langganan pelanggan, Anda menangani `subscribe-success` notifikasi dengan membuat atau mengambil kunci API Gateway API untuk pelanggan, mengaitkan AWS Marketplace-provisioned pelanggan `customerId` dengan kunci API, dan kemudian menambahkan kunci API ke paket penggunaan Anda.

Ketika permintaan langganan pelanggan selesai, aplikasi portal pengembang harus menyajikan kunci API terkait kepada pelanggan dan memberi tahu pelanggan bahwa kunci API harus disertakan dalam `x-api-key` header dalam permintaan ke APIs.

Saat pelanggan membatalkan langganan paket penggunaan, AWS Marketplace kirimkan `unsubscribe-success` pemberitahuan ke topik SNS. Untuk menyelesaikan proses berhenti berlangganan pelanggan, Anda menangani `unsubscribe-success` notifikasi dengan menghapus kunci API pelanggan dari paket penggunaan.

Otorisasi pelanggan untuk mengakses paket penggunaan

Untuk mengotorisasi akses ke paket penggunaan Anda bagi pelanggan tertentu, gunakan API Gateway API untuk mengambil atau membuat kunci API bagi pelanggan dan menambahkan kunci API ke paket penggunaan.

Contoh berikut menunjukkan cara memanggil API Gateway REST API untuk membuat kunci API baru dengan AWS Marketplace `customerId` nilai tertentu (*MARKETPLACE_CUSTOMER_ID*).

```
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"
}
```

Contoh berikut menunjukkan cara mendapatkan kunci API dengan AWS Marketplace `customerId` nilai tertentu (*MARKETPLACE_CUSTOMER_ID*).

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Untuk menambahkan kunci API ke paket penggunaan, buat kunci [UsagePlanKey](#) dengan API untuk paket penggunaan yang relevan. Contoh berikut menunjukkan cara melakukannya menggunakan API Gateway REST API, di mana `n371pt` ID paket penggunaan dan `q5ugs7qjjh` merupakan contoh API yang `keyId` dikembalikan dari contoh sebelumnya.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
```

```
"keyId": "q5ugs7qjjh",  
"keyType": "API_KEY"  
}
```

Kaitkan pelanggan dengan kunci API

Anda harus memperbarui `customerId` kolom ke ID AWS Marketplace pelanggan pelanggan.

[ApiKey](#) Ini mengaitkan kunci API dengan AWS Marketplace pelanggan, yang memungkinkan pengukuran dan penagihan untuk pembeli. Contoh kode berikut memanggil API Gateway REST API untuk melakukannya.

```
PATCH /apikeys/q5ugs7qjjh  
Host: apigateway.region.amazonaws.com  
Authorization: ...  
  
{  
  "patchOperations" : [{  
    "path" : "/customerId",  
    "value" : "MARKETPLACE_CUSTOMER_ID",  
    "op" : "replace"  
  }]  
}
```

Lindungi REST Anda APIs di API Gateway

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti pengguna jahat atau lonjakan lalu lintas. Anda dapat melindungi API menggunakan strategi seperti membuat sertifikat SSL, mengonfigurasi firewall aplikasi web, menetapkan target pelambatan, dan hanya mengizinkan akses ke API Anda dari Virtual Private Cloud (VPC). Di bagian ini Anda dapat mempelajari cara mengaktifkan kemampuan ini menggunakan API Gateway.

Topik

- [Cara mengaktifkan otentikasi TLS timbal balik untuk REST Anda APIs di API Gateway](#)
- [Buat dan konfigurasi sertifikat SSL untuk otentikasi backend di API Gateway](#)
- [Gunakan AWS WAF untuk melindungi REST Anda APIs di API Gateway](#)
- [Throttle request ke REST APIs Anda untuk throughput yang lebih baik di API Gateway](#)
- [REST pribadi APIs di API Gateway](#)

Cara mengaktifkan otentikasi TLS timbal balik untuk REST Anda APIs di API Gateway

Mutual TLS otentikasi membutuhkan otentikasi dua arah antara klien dan server. Dengan TLS bersama, klien harus menunjukkan sertifikat X.509 untuk memverifikasi identitas mereka untuk mengakses API Anda. Mutual TLS adalah persyaratan umum untuk Internet of Things (IoT) business-to-business dan aplikasi.

Anda dapat menggunakan TLS timbal balik bersama dengan [operasi otorisasi dan otentikasi](#) lain yang didukung API Gateway. API Gateway meneruskan sertifikat yang disediakan klien kepada otorisasi Lambda dan integrasi backend.

Important

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus dengan TLS timbal balik, nonaktifkan titik `execute-api` akhir default. Untuk mempelajari selengkapnya, lihat [the section called “Nonaktifkan titik akhir default”](#).

Topik

- [Prasyarat untuk TLS timbal balik](#)
- [Mengkonfigurasi TLS timbal balik untuk nama domain khusus](#)
- [Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik](#)
- [Memperbarui truststore Anda](#)
- [Nonaktifkan TLS timbal balik](#)
- [Memecahkan masalah TLS timbal balik untuk REST API Anda](#)

Prasyarat untuk TLS timbal balik

Untuk mengkonfigurasi TLS timbal balik yang Anda butuhkan:

- Nama domain kustom Regional
- Setidaknya satu sertifikat dikonfigurasi AWS Certificate Manager untuk nama domain kustom Anda

- Truststore dikonfigurasi dan diunggah ke Amazon S3

Nama domain khusus

Untuk mengaktifkan TLS timbal balik untuk REST API, Anda harus mengonfigurasi nama domain khusus untuk API Anda. Anda dapat mengaktifkan TLS timbal balik untuk nama domain khusus, dan kemudian memberikan nama domain khusus kepada klien. Untuk mengakses API dengan menggunakan nama domain kustom yang mengaktifkan TLS bersama, klien harus menunjukkan sertifikat yang Anda percayai dalam permintaan API. Anda dapat menemukan informasi lebih lanjut [di the section called “Nama domain kustom”](#).

Menggunakan sertifikat AWS Certificate Manager yang dikeluarkan

Anda dapat meminta sertifikat tepercaya publik langsung dari ACM atau mengimpor sertifikat publik atau yang ditandatangani sendiri. Untuk menyiapkan sertifikat di ACM, buka [ACM](#). Jika Anda ingin mengimpor sertifikat, lanjutkan membaca di bagian berikut.

Menggunakan impor atau AWS Private Certificate Authority sertifikat

Untuk menggunakan sertifikat yang diimpor ke ACM atau sertifikat dari TLS AWS Private Certificate Authority bersama, API Gateway memerlukan yang `ownershipVerificationCertificate` dikeluarkan oleh ACM. Sertifikat kepemilikan ini hanya digunakan untuk memverifikasi bahwa Anda memiliki izin untuk menggunakan nama domain. Ini tidak digunakan untuk jabat tangan TLS. Jika Anda belum memiliki `ownershipVerificationCertificate`, pergi ke <https://console.aws.amazon.com/acm/> untuk mengaturnya.

Anda harus menjaga sertifikat ini tetap berlaku selama masa pakai nama domain Anda. Jika sertifikat kedaluwarsa dan perpanjangan otomatis gagal, semua pembaruan pada nama domain akan dikunci. Anda perlu memperbarui `ownershipVerificationCertificateArn` dengan valid `ownershipVerificationCertificate` sebelum Anda dapat membuat perubahan lainnya. Ini `ownershipVerificationCertificate` tidak dapat digunakan sebagai sertifikat server untuk domain TLS timbal balik lainnya di API Gateway. Jika sertifikat langsung diimpor kembali ke ACM, penerbit harus tetap sama.

Mengkonfigurasi truststore Anda

Truststores adalah file teks dengan ekstensi `.pem` file. Mereka adalah daftar sertifikat tepercaya dari Otoritas Sertifikat. Untuk menggunakan TLS timbal balik, buat truststore sertifikat X.509 yang Anda percayai untuk mengakses API Anda.

Anda harus menyertakan rantai kepercayaan lengkap, mulai dari sertifikat CA penerbitan, hingga sertifikat CA root, di truststore Anda. API Gateway menerima sertifikat klien yang dikeluarkan oleh CA mana pun yang ada dalam rantai kepercayaan. Sertifikat dapat dari otoritas sertifikat publik atau swasta. Sertifikat dapat memiliki panjang rantai maksimum empat. Anda juga dapat memberikan sertifikat yang ditandatangani sendiri. Algoritma berikut didukung di truststore:

- SHA-256 atau lebih kuat
- RSA-2048 atau lebih kuat
- ECDSA-256 atau ECDSA-384

API Gateway memvalidasi sejumlah properti sertifikat. Anda dapat menggunakan otorisasi Lambda untuk melakukan pemeriksaan tambahan saat klien memanggil API, termasuk memeriksa apakah sertifikat telah dicabut. API Gateway memvalidasi properti berikut:

Validasi	Deskripsi
Sintaks X.509	Sertifikat harus memenuhi persyaratan sintaks X.509.
Integritas	Konten sertifikat tidak boleh diubah dari yang ditandatangani oleh otoritas sertifikat dari truststore.
Validitas	Masa berlaku sertifikat harus terkini.
Nama rantai/rantai kunci	Nama dan subjek sertifikat harus membentuk rantai yang tidak terputus. Sertifikat dapat memiliki panjang rantai maksimum empat.

Unggah truststore ke bucket Amazon S3 dalam satu file

Berikut ini adalah contoh dari apa file.pem mungkin terlihat seperti.

Example sertifikat.pem

```
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
...
```

AWS CLI Perintah [cp](#) berikut diunggah `certificates.pem` ke bucket Amazon S3 Anda:

```
aws s3 cp certificates.pem s3://bucket-name
```

Mengkonfigurasi TLS timbal balik untuk nama domain khusus

Untuk mengonfigurasi TLS bersama untuk REST API, Anda harus menggunakan nama domain kustom Regional untuk API Anda, dengan kebijakan TLS_1_2 keamanan. Untuk informasi selengkapnya tentang memilih kebijakan keamanan, lihat [the section called “Pilih kebijakan keamanan”](#).

Note

Mutual TLS tidak didukung untuk pribadi APIs.

Setelah Anda mengunggah truststore Anda ke Amazon S3, Anda dapat mengonfigurasi nama domain kustom Anda untuk menggunakan TLS bersama. Berikut ini [create-domain-name](#) membuat nama domain kustom dengan TLS timbal balik:

```
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/123456789012-1234-1234-12345678 \  
  --endpoint-configuration types=REGIONAL \  
  --security-policy TLS_1_2 \  
  --mutual-tls-authentication truststoreUri=s3://bucket-name/key-name
```

Setelah membuat nama domain, Anda harus mengonfigurasi catatan DNS dan pemetaan basepath untuk operasi API. Untuk mempelajari selengkapnya, lihat [Siapkan nama domain kustom Regional di API Gateway](#).

Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik

Untuk menjalankan API dengan TLS timbal balik diaktifkan, klien harus menunjukkan sertifikat tepercaya dalam permintaan API. Saat klien mencoba menjalankan API Anda, API Gateway mencari penerbit sertifikat klien di truststore Anda. Agar API Gateway dapat melanjutkan permintaan, penerbit sertifikat dan rantai kepercayaan lengkap hingga sertifikat CA root harus ada di truststore Anda.

Contoh `curl` perintah berikut mengirimkan permintaan ke `api.example.com`, yang termasuk `my-cert.pem` dalam permintaan. `my-key.key` adalah kunci pribadi untuk sertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

API Anda dipanggil hanya jika truststore Anda mempercayai sertifikat. Kondisi berikut akan menyebabkan API Gateway gagal dalam jabat tangan TLS dan menolak permintaan dengan kode 403 status. Jika sertifikat Anda:

- tidak dipercaya
- kedaluwarsa
- tidak menggunakan algoritma yang didukung

Note

API Gateway tidak memverifikasi apakah sertifikat telah dicabut.

Memperbarui truststore Anda

Untuk memperbarui sertifikat di truststore Anda, unggah bundel sertifikat baru ke Amazon S3. Kemudian, Anda dapat memperbarui nama domain kustom Anda untuk menggunakan sertifikat yang diperbarui.

Gunakan versi [Amazon S3 untuk mempertahankan beberapa versi](#) truststore Anda. Saat memperbarui nama domain khusus untuk menggunakan versi truststore baru, API Gateway mengembalikan peringatan jika sertifikat tidak valid.

API Gateway menghasilkan peringatan sertifikat hanya ketika Anda memperbarui nama domain Anda. API Gateway tidak memberi tahu Anda jika sertifikat yang diunggah sebelumnya kedaluwarsa.

[update-domain-name](#) Perintah berikut memperbarui nama domain kustom untuk menggunakan versi truststore baru:

```
aws apigateway update-domain-name \  
  --domain-name api.example.com \  
  --patch-operations op='replace',path='/mutualTlsAuthentication/  
truststoreVersion',value='abcdef123'
```

Nonaktifkan TLS timbal balik

Berikut ini [update-domain-name](#) menonaktifkan TLS timbal balik:

```
aws apigateway update-domain-name \  
  --domain-name api.example.com \  
  --patch-operations op='replace',path='/mutualTlsAuthentication/  
truststoreUri',value=''
```

Memecahkan masalah TLS timbal balik untuk REST API Anda

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat mengaktifkan TLS timbal balik.

Memecahkan masalah peringatan sertifikat

Saat membuat nama domain khusus dengan TLS timbal balik, API Gateway mengembalikan peringatan jika sertifikat di truststore tidak valid. Ini juga dapat terjadi saat memperbarui nama domain khusus untuk menggunakan truststore baru. Peringatan menunjukkan masalah dengan sertifikat dan subjek sertifikat yang menghasilkan peringatan. Mutual TLS masih diaktifkan untuk API Anda, tetapi beberapa klien mungkin tidak dapat mengakses API Anda.

Anda harus memecahkan kode sertifikat di truststore Anda untuk mengidentifikasi sertifikat mana yang menghasilkan peringatan. Anda dapat menggunakan alat seperti `openssl` untuk memecahkan kode sertifikat dan mengidentifikasi subjek mereka.

Perintah berikut menampilkan isi sertifikat, termasuk subjeknya:

```
openssl x509 -in certificate.crt -text -noout
```

Perbarui atau hapus sertifikat yang menghasilkan peringatan, lalu unggah truststore baru ke Amazon S3. Setelah mengunggah truststore baru, perbarui nama domain kustom Anda untuk menggunakan truststore baru.

Memecahkan masalah konflik nama domain

Kesalahan "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." berarti beberapa Otoritas Sertifikat telah mengeluarkan sertifikat untuk domain ini. Untuk setiap subjek dalam sertifikat, hanya ada satu penerbit di API Gateway untuk domain TLS bersama. Anda harus mendapatkan semua sertifikat Anda untuk subjek itu melalui satu penerbit. Jika masalahnya adalah dengan sertifikat yang tidak Anda kendalikan tetapi Anda dapat membuktikan kepemilikan nama domain, [hubungi Dukungan](#) untuk membuka tiket.

Memecahkan masalah pesan status nama domain

PENDING_CERTIFICATE_REIMPORT: Ini berarti Anda mengimpor ulang sertifikat ke ACM dan validasi gagal karena sertifikat baru memiliki SAN (nama alternatif subjek) yang tidak tercakup oleh ownershipVerificationCertificate atau subjek atau SANs dalam sertifikat tidak mencakup nama domain. Sesuatu mungkin dikonfigurasi dengan tidak benar atau sertifikat yang tidak valid diimpor. Anda perlu mengimpor ulang sertifikat yang valid ke ACM. Untuk informasi selengkapnya tentang validasi, lihat [Memvalidasi kepemilikan domain](#).

PENDING_OWNERSHIP_VERIFICATION: Ini berarti sertifikat Anda yang telah diverifikasi sebelumnya telah kedaluwarsa dan ACM tidak dapat memperbaruinya secara otomatis. Anda perlu memperbarui sertifikat atau meminta sertifikat baru. Informasi lebih lanjut tentang perpanjangan sertifikat dapat ditemukan di panduan perpanjangan sertifikat [terkelola pemecahan masalah ACM](#).

Memecahkan masalah sertifikat yang dikembalikan salah

Saat memigrasikan sertifikat khusus dari nama domain yang memenuhi syarat penuh (FQDN) ke nama domain pelanggan wildcard, API Gateway mungkin mengembalikan sertifikat untuk FQDN, bukan nama domain wildcard.

Perintah berikut menampilkan sertifikat mana yang dikembalikan oleh API Gateway:

```
openssl s_client -connect hostname:port
```

Jika sertifikat yang dihasilkan adalah untuk FQDN, [hubungi Dukungan](#) untuk membuka tiket.

Buat dan konfigurasi sertifikat SSL untuk otentikasi backend di API Gateway

Anda dapat menggunakan API Gateway untuk menghasilkan sertifikat SSL dan kemudian menggunakan kunci publiknya di backend untuk memverifikasi bahwa permintaan HTTP ke sistem backend Anda berasal dari API Gateway. Hal ini memungkinkan backend HTTP Anda untuk mengontrol dan hanya menerima permintaan yang berasal dari Amazon API Gateway, bahkan jika backend dapat diakses publik.

Note

Beberapa server backend mungkin tidak mendukung otentikasi klien SSL seperti yang dilakukan API Gateway dan dapat mengembalikan kesalahan sertifikat SSL. Untuk daftar server backend yang tidak kompatibel, lihat [the section called “Catatan penting”](#)

Sertifikat SSL yang dihasilkan oleh API Gateway ditandatangani sendiri, dan hanya kunci publik sertifikat yang terlihat di konsol API Gateway atau melalui APIs

Topik

- [Buat sertifikat klien menggunakan konsol API Gateway](#)
- [Konfigurasi API untuk menggunakan sertifikat SSL](#)
- [Uji pemanggilan untuk memverifikasi konfigurasi sertifikat klien](#)
- [Konfigurasi server HTTPS backend untuk memverifikasi sertifikat klien](#)
- [Putar sertifikat klien yang kedaluwarsa](#)
- [Otoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP di API Gateway](#)

Buat sertifikat klien menggunakan konsol API Gateway

1. Buka konsol API Gateway di <https://console.aws.amazon.com/apigateway/>.
2. Pilih REST atau WebSocket API.
3. Di panel navigasi utama, pilih Sertifikat klien.
4. Dari halaman Sertifikat klien, pilih Hasilkan sertifikat.

5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Pilih Hasilkan sertifikat untuk menghasilkan sertifikat. API Gateway menghasilkan sertifikat baru dan mengembalikan GUID sertifikat baru, bersama dengan kunci publik yang dikodekan PEM.

Anda sekarang siap untuk mengkonfigurasi API untuk menggunakan sertifikat.

Konfigurasi API untuk menggunakan sertifikat SSL

Instruksi ini mengasumsikan bahwa Anda sudah selesai [Buat sertifikat klien menggunakan konsol API Gateway](#).

1. Di konsol API Gateway, buat atau buka REST atau WebSocket API yang ingin Anda gunakan sertifikat klien. Pastikan bahwa API telah diterapkan ke sebuah panggung.
2. Di panel navigasi utama, pilih Tahapan.
3. Di bagian Detail tahap, pilih Edit.
4. Untuk sertifikat Klien, pilih sertifikat.
5. Pilih Simpan perubahan.

Setelah sertifikat dipilih untuk API dan disimpan, API Gateway menggunakan sertifikat untuk semua panggilan ke integrasi HTTP di API Anda.

Uji pemanggilan untuk memverifikasi konfigurasi sertifikat klien

1. Pilih metode REST API. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Uji.
2. Untuk sertifikat Klien, pilih sertifikat.
3. Pilih Uji.

API Gateway menyajikan sertifikat SSL yang dipilih untuk backend HTTP untuk mengautentikasi API.

Konfigurasi server HTTPS backend untuk memverifikasi sertifikat klien

Instruksi ini mengasumsikan bahwa Anda sudah menyelesaikan [Buat sertifikat klien menggunakan konsol API Gateway](#) dan mengunduh salinan sertifikat klien. Anda dapat mengunduh sertifikat klien dengan memanggil [clientcertificate:by-id](#) API Gateway REST API atau [get-client-certificate](#) dari AWS CLI.

Sebelum mengonfigurasi server HTTPS backend untuk memverifikasi sertifikat SSL klien API Gateway, Anda harus telah memperoleh kunci pribadi yang dikodekan PEM dan sertifikat sisi server yang disediakan oleh otoritas sertifikat tepercaya.

Jika nama domain `servermyserver.mydomain.com`, nilai CNAME sertifikat server harus `myserver.mydomain.com` atau `*.mydomain.com`.

Otoritas sertifikat yang didukung termasuk [Let's Encrypt](#) atau salah satu dari [the section called "Otoritas sertifikat yang didukung untuk integrasi proxy HTTP dan HTTP"](#)

Sebagai contoh, misalkan file sertifikat klien adalah `apig-cert.pem` dan kunci pribadi server dan file sertifikat adalah `server-key.pem` dan `server-cert.pem`, masing-masing. Untuk server Node.js di backend, Anda dapat mengonfigurasi server yang mirip dengan yang berikut ini:

```
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);
```

Untuk aplikasi node- [express](#), Anda dapat menggunakan [client-certificate-auth](#) modul untuk mengautentikasi permintaan klien dengan sertifikat yang dikodekan PEM.

Untuk server HTTPS lainnya, lihat dokumentasi untuk server.

Putar sertifikat klien yang kedaluwarsa

Sertifikat klien yang dihasilkan oleh API Gateway berlaku selama 365 hari. Anda harus memutar sertifikat sebelum sertifikat klien pada tahap API kedaluwarsa untuk menghindari waktu henti API.

Putar sertifikat klien yang kedaluwarsa menggunakan Konsol Manajemen AWS

Prosedur berikut menunjukkan cara memutar sertifikat klien di konsol untuk API yang digunakan sebelumnya.

1. Di panel navigasi utama, pilih Sertifikat klien.
2. Dari panel Sertifikat klien, pilih Menghasilkan sertifikat.
3. Buka API yang ingin Anda gunakan sertifikat klien.
4. Pilih Tahapan di bawah API yang dipilih dan kemudian pilih panggung.
5. Di bagian Detail tahap, pilih Edit.
6. Untuk sertifikat Klien, pilih sertifikat baru.
7. Untuk menyimpan pengaturan, pilih Simpan perubahan.

Putar sertifikat klien yang kedaluwarsa menggunakan AWS CLI

[Anda dapat memeriksa tanggal kedaluwarsa sertifikat dengan memanggil `clientCertificate:by-ID` API Gateway REST API AWS CLI atau perintah dan memeriksa properti `ExpirationDate` `get-client-certificate` yang dikembalikan.](#)

Untuk memutar sertifikat klien, lakukan hal berikut:

1. Buat sertifikat klien baru dengan memanggil [clientcertificate:generate](#) API Gateway REST API atau perintah dari AWS CLI [generate-client-certificate](#). Dalam tutorial ini, kita berasumsi bahwa ID sertifikat klien baru adalah `ndiqef`.
2. Perbarui server backend untuk menyertakan sertifikat klien baru. Jangan menghapus sertifikat klien yang ada.

Beberapa server mungkin memerlukan restart untuk menyelesaikan pembaruan. Konsultasikan dokumentasi server untuk melihat apakah Anda harus me-restart server selama pembaruan.

3. Perbarui tahap API untuk menggunakan sertifikat klien baru dengan memanggil [stage:update](#) API Gateway REST API, dengan ID sertifikat klien baru (`:`): `ndiqef`

```
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"
  }
]
}

```

Anda juga dapat menggunakan perintah [update-stage](#).

Jika Anda menggunakan WebSocket API, gunakan perintah `apigatewayv2` [update-stage](#).

4. Perbarui server backend untuk menghapus sertifikat lama.
5. Hapus sertifikat lama dari API Gateway dengan memanggil [clientcertificate:delete](#) API Gateway REST API, dengan menentukan `clientCertificateId` () a1b2c3 sertifikat lama:

```
DELETE /clientcertificates/a1b2c3
```

Anda juga dapat memanggil [delete-client-certificate](#) perintah:

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Otoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP di API Gateway

Daftar berikut menunjukkan otoritas sertifikat yang didukung oleh API Gateway untuk HTTP, proxy HTTP, dan integrasi pribadi.

```

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: atotrustedroot2011
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: chunghwaepkirootca
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: dstrootcax3
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: microseceszignorootca2009
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: mozillacert120.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: mozillacert121.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: mozillacert122.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: mozillacert123.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: mozillacert140.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: mozillacert141.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: mozillacert142.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: mozillacert143.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: mozillacert3.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: mozillacert30.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
Alias name: mozillacert31.pem
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: mozillacert4.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: mozillacert40.pem
SHA1: 80:25:EF:F4:6E:70:C8:D4:72:24:65:84:FE:40:3B:8A:8D:6A:DB:F5
SHA256:
8D:A0:84:FC:F9:9C:E0:77:22:F8:9B:32:05:93:98:06:FA:5C:B8:11:E1:C8:13:F6:A1:08:C7:D3:36:B3:40:8
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: mozillacert5.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: mozillacert50.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
Alias name: mozillacert51.pem
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: mozillacert6.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: mozillacert60.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: mozillacert7.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: mozillacert70.pem
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: 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: mozillacert8.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: mozillacert80.pem
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: 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: mozillacert9.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: mozillacert90.pem
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: 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: quovadisrootcalg3
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: soneraiclass1ca
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: soneraiclass2ca
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: soneraiclass2rootca
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: sslcomevrootcertificationauthorityrsa2
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:B6:5F
Alias name: starfielddrootcertificateauthorityg2
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:F9
Alias name: starfielddrootg2ca
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:F9
Alias name: starfielddservicesrootcertificateauthorityg2
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:B0
Alias name: starfielddservicesrootg2ca
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:B0
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:90
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:90
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:14:30
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:D0
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:D0
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:F0
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: trustcorecal
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: trustcorrootcertcal
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: trustisfpsrootca
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: tubitakkamusmsslkoksertifikasisurum1
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: utndatacorpsgccca
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: verisignc1g1.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: verisignc1g2.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: verisignc1g3.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: verisignc1g6.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: xrapglobalca
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: xrapglobalcaroot
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

```

Gunakan AWS WAF untuk melindungi REST Anda APIs di API Gateway

AWS WAF adalah firewall aplikasi web yang membantu melindungi aplikasi web dan APIs dari serangan. Ini memungkinkan Anda untuk mengonfigurasi seperangkat aturan yang disebut daftar

kontrol akses web (web ACL) yang memungkinkan, memblokir, atau menghitung permintaan web berdasarkan aturan dan kondisi keamanan web yang dapat disesuaikan yang Anda tentukan. Untuk informasi selengkapnya, lihat [Cara AWS WAF Kerja](#).

Anda dapat menggunakan AWS WAF untuk melindungi API Gateway REST API Anda dari eksploitasi web umum, seperti injeksi SQL dan serangan cross-site scripting (XSS). Ini dapat memengaruhi ketersediaan dan kinerja API, membahayakan keamanan, atau mengkonsumsi sumber daya yang berlebihan. Misalnya, Anda dapat membuat aturan untuk mengizinkan atau memblokir permintaan dari rentang alamat IP tertentu, permintaan dari blok CIDR, permintaan yang berasal dari negara atau wilayah tertentu, permintaan yang berisi kode SQL berbahaya, atau permintaan yang berisi skrip berbahaya.

Anda juga dapat membuat aturan yang cocok dengan string tertentu atau pola ekspresi reguler di header HTTP, metode, string kueri, URI, dan badan permintaan (terbatas pada 64 KB pertama). Selain itu, Anda dapat membuat aturan untuk memblokir serangan dari agen pengguna tertentu, bot buruk, dan pencakar konten. Misalnya, Anda dapat menggunakan aturan berbasis tarif untuk menentukan jumlah permintaan web yang diizinkan oleh setiap IP klien dalam periode 5 menit yang terus diperbarui.

Important

AWS WAF adalah garis pertahanan pertama Anda terhadap eksploitasi web. [Bila AWS WAF diaktifkan pada API, AWS WAF aturan dievaluasi sebelum fitur kontrol akses lainnya, seperti kebijakan sumber daya, kebijakan IAM, otorisasi Lambda, dan otorisasi Amazon Cognito.](#) Misalnya, jika AWS WAF memblokir akses dari blok CIDR yang diizinkan oleh kebijakan sumber daya, AWS WAF diutamakan dan kebijakan sumber daya tidak dievaluasi.

AWS WAF Untuk mengaktifkan API Anda, Anda perlu melakukan hal berikut:

1. Gunakan AWS WAF konsol, AWS SDK, atau CLI untuk membuat ACL web yang berisi kombinasi aturan terkelola yang diinginkan dan aturan AWS WAF kustom Anda sendiri. Untuk informasi selengkapnya, lihat [Memulai AWS WAF](#) dan [daftar kontrol akses Web \(web ACLs\)](#).

Important

API Gateway memerlukan ACL AWS WAFV2 web untuk aplikasi Regional atau ACL AWS WAF Classic Regional web.

2. Kaitkan ACL AWS WAF web dengan tahap API. Anda dapat melakukannya dengan menggunakan AWS WAF konsol, AWS SDK, CLI, atau dengan menggunakan konsol API Gateway.

Untuk mengaitkan ACL AWS WAF web dengan tahap API Gateway API menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mengaitkan ACL AWS WAF web dengan tahap API Gateway API yang ada, gunakan langkah-langkah berikut:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih panggung.
4. Di bagian Detail tahap, pilih Edit.
5. Di bawah Firewall aplikasi Web (AWS WAF), pilih ACL web Anda.

Jika Anda menggunakan AWS WAFV2, pilih ACL AWS WAFV2 web untuk aplikasi Regional. ACL web dan AWS WAFV2 sumber daya lain yang digunakannya harus berada di Wilayah yang sama dengan API Anda.

Jika Anda menggunakan AWS WAF Classic Regional, pilih ACL web Regional.

6. Pilih Simpan perubahan.

Kaitkan ACL AWS WAF web dengan tahap API Gateway API menggunakan AWS CLI

[associate-web-acl](#) Perintah berikut mengaitkan ACL AWS WAFV2 web untuk aplikasi Regional dengan tahap API Gateway API yang ada:

```
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
```

[associate-web-acl](#) Perintah berikut mengaitkan ACL AWS WAF Classic Regional web dengan tahap API Gateway API yang ada:

```
aws waf-regional associate-web-acl \  

```

```
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \  
--resource-arn 'arn:aws:apigateway:{region}:/restapis/4wk1k4onj3/stages/prod'
```

Kaitkan ACL AWS WAF web dengan tahap API menggunakan AWS WAF REST API

Untuk menggunakan AWS WAFV2 REST API untuk mengaitkan ACL AWS WAFV2 web untuk aplikasi Regional dengan tahap API Gateway API yang ada, gunakan perintah [AssociateWebACL](#), seperti pada contoh berikut:

```
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'  
)
```

Untuk menggunakan AWS WAF REST API untuk mengaitkan ACL AWS WAF Classic Regional web dengan tahap API Gateway API yang ada, gunakan perintah [AssociateWebACL](#), seperti pada contoh berikut:

```
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'  
)
```

Throttle request ke REST APIs Anda untuk throughput yang lebih baik di API Gateway

Anda dapat mengonfigurasi pembatasan dan kuota APIs untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Baik throttle dan kuota diterapkan atas dasar upaya terbaik dan harus dianggap sebagai target daripada plafon permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritme token bucket, tempat token diperhitungkan untuk permintaan. Secara khusus, API Gateway memeriksa tarif dan ledakan pengiriman permintaan terhadap semua yang ada APIs di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batas dikuasai dalam beberapa kasus.

Jika pengiriman permintaan melebihi tingkat permintaan kondisi tunak dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima tanggapan 429 Too Many Requests kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan atau metode API untuk meningkatkan kinerja keseluruhan APIs di semua akun Anda. Atau, Anda dapat mengaktifkan paket penggunaan untuk mengatur pembatasan pada pengiriman permintaan klien berdasarkan tarif permintaan dan kuota yang ditentukan.

Topik

- [Bagaimana pengaturan batas pelambatan diterapkan di API Gateway](#)
- [Pelambatan tingkat akun per Wilayah](#)
- [Mengonfigurasi target pelambatan level API dan level tahap dalam rencana penggunaan](#)
- [Mengkonfigurasi target pelambatan tingkat tahap](#)
- [Mengonfigurasi target pelambatan tingkat metode dalam rencana penggunaan](#)

Bagaimana pengaturan batas pelambatan diterapkan di API Gateway

Sebelum mengonfigurasi setelan throttle dan kuota untuk API, sebaiknya pahami jenis setelan terkait throttling untuk API Anda dan bagaimana API Gateway menerapkannya.

Amazon API Gateway menyediakan empat tipe dasar pengaturan terkait pelambatan:

- AWS batas pembatasan diterapkan di semua akun dan klien di suatu Wilayah. Pengaturan batas ini ada untuk mencegah API Anda—dan akun Anda—kewalahan oleh terlalu banyak permintaan. Batasan ini ditetapkan oleh AWS dan tidak dapat diubah oleh pelanggan.
- Batas per akun diterapkan ke semua akun APIs di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan batas waktu APIs yang lebih pendek dan muatan yang lebih kecil. [Untuk meminta peningkatan batas pembatasan tingkat akun per Wilayah, hubungi Pusat Dukungan.AWS](#) Untuk informasi

selengkapnya, lihat [Kuota dan catatan penting](#). Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.

- Batas pelambatan per API, per tahap diterapkan pada tingkat metode API untuk suatu tahap. Anda dapat mengonfigurasi pengaturan yang sama untuk semua metode, atau mengonfigurasi pengaturan throttle yang berbeda untuk setiap metode. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.
- Batas pembatasan per klien diterapkan pada klien yang menggunakan kunci API yang terkait dengan paket penggunaan Anda sebagai pengenalan klien. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas per akun.

API Gateway menerapkan setelan terkait throttling Anda dalam urutan berikut:

1. [Batas pembatasan per klien atau per-metode yang Anda tetapkan untuk tahap API dalam paket penggunaan](#)
2. [Batas pembatasan per metode yang Anda tetapkan untuk tahap API](#)
3. [Pelambatan tingkat akun per Wilayah](#)
4. AWS Pelambatan regional

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan kondisi tunak per detik (RPS) APIs di semua AWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua APIs dalam AWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target pengiriman permintaan bersamaan yang akan dipenuhi oleh API Gateway sebelum mengembalikan respons 429 Too Many Requests kesalahan. Untuk informasi lebih lanjut tentang pembatasan kuota, lihat [Kuota dan catatan penting](#).

Mengonfigurasi target pelambatan level API dan level tahap dalam rencana penggunaan

Dalam [paket penggunaan](#), Anda dapat menetapkan target pelambatan per metode untuk semua metode di API atau level tahap. Anda dapat menentukan laju pelambatan, yang merupakan tingkat, dalam permintaan per detik, token tersebut ditambahkan ke keranjang token. Anda juga dapat menentukan burst throttling, yang merupakan kapasitas bucket token.

Anda dapat menggunakan AWS CLI, SDKs, dan Konsol Manajemen AWS untuk membuat rencana penggunaan. Untuk informasi selengkapnya tentang cara membuat paket penggunaan, lihat [???](#).

Mengonfigurasi target pelambatan tingkat tahap

Anda dapat menggunakan AWS CLI, SDKs, dan Konsol Manajemen AWS untuk membuat target pelambatan tingkat tahap.

Untuk informasi selengkapnya tentang cara menggunakan Konsol Manajemen AWS untuk membuat target pelambatan tingkat tahap, lihat. [??? Untuk informasi selengkapnya tentang cara menggunakan AWS CLI untuk membuat target pelambatan tingkat tahap, lihat create-stage.](#)

Mengonfigurasi target pelambatan tingkat metode dalam rencana penggunaan

Anda dapat menetapkan target pelambatan tambahan pada tingkat metode dalam Rencana Penggunaan seperti yang ditunjukkan pada. [Buat rencana penggunaan](#) Di konsol API Gateway, ini diatur dengan menentukan `Resource=<resource>`, `Method=<method>` dalam pengaturan Configure Method Throttling. Misalnya, [PetStoremisalnya](#), Anda dapat menentukan `Resource=/pets`, `Method=GET`.

REST pribadi APIs di API Gateway

API pribadi adalah REST API yang hanya dapat dipanggil dari dalam VPC Amazon. Anda dapat mengakses API Anda menggunakan [antarmuka VPC endpoint](#), yang merupakan antarmuka jaringan endpoint yang Anda buat di VPC Anda. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses AWS layanan secara pribadi dengan menggunakan alamat IP pribadi.

Anda juga dapat menggunakan Direct Connect untuk membuat sambungan dari jaringan lokal ke Amazon VPC dan kemudian mengakses API pribadi Anda melalui koneksi tersebut. Dalam semua kasus, lalu lintas ke API pribadi Anda menggunakan koneksi aman dan terisolasi dari internet publik. Lalu lintas tidak meninggalkan jaringan Amazon.

Praktik terbaik untuk pribadi APIs

Sebaiknya gunakan praktik terbaik berikut saat membuat API pribadi:

- Gunakan satu titik akhir VPC untuk mengakses beberapa pribadi. APIs Ini mengurangi jumlah titik akhir VPC yang mungkin Anda butuhkan.

- Kaitkan titik akhir VPC Anda ke API Anda. Ini membuat catatan DNS alias Route 53 dan menyederhanakan pemanggilan API pribadi Anda.
- Aktifkan DNS pribadi untuk VPC Anda. Saat Anda mengaktifkan DNS pribadi untuk VPC Anda, Anda dapat menjalankan API Anda dalam VPC tanpa melewati header atau. Host `x-apigw-api-id`

Jika Anda mengaktifkan DNS pribadi, Anda tidak dapat mengakses titik akhir default untuk publik. APIs Untuk mengakses titik akhir default untuk publik APIs, Anda dapat menonaktifkan DNS pribadi, membuat zona host pribadi untuk setiap API pribadi di VPC Anda, dan kemudian menyediakan catatan yang diperlukan di Route 53. Ini memungkinkan API pribadi Anda untuk diselesaikan sementara Anda masih dapat memanggil titik akhir default publik dari VPC Anda. Untuk informasi selengkapnya, lihat [Membuat zona host pribadi](#).

- Batasi akses ke API pribadi Anda ke titik akhir tertentu VPCs atau VPC. Tambahkan `aws:SourceVpc` atau `aws:SourceVpce` kondisi ke kebijakan sumber daya API Anda untuk membatasi akses.
- Untuk perimeter data yang paling aman, Anda dapat membuat kebijakan titik akhir VPC. Ini mengontrol akses ke titik akhir VPC yang dapat menjalankan API pribadi Anda.

Pertimbangan untuk pribadi APIs

Pertimbangan berikut dapat memengaruhi penggunaan pribadi APIs Anda:

- Hanya REST APIs yang didukung.
- Anda tidak dapat mengonversi API pribadi ke API yang dioptimalkan tepi.
- Pribadi APIs hanya mendukung TLS 1.2. Versi TLS sebelumnya tidak didukung.
- Jika Anda membuat permintaan menggunakan protokol HTTP/2, permintaan diberlakukan untuk menggunakan protokol HTTP/1.1.
- Anda tidak dapat mengatur jenis alamat IP untuk pribadi APIs agar hanya mengizinkan IPv4 alamat menjalankan API pribadi Anda. Hanya dualstack yang didukung. Untuk informasi selengkapnya, lihat [the section called “Jenis alamat IP untuk REST APIs di API Gateway”](#).
- Untuk mengirim lalu lintas menggunakan API pribadi, Anda dapat menggunakan semua jenis alamat IP yang didukung oleh Amazon VPC. Anda dapat mengirim dualstack dan IPv6 lalu lintas dengan mengonfigurasi pengaturan pada titik akhir VPC Anda. Anda tidak dapat memodifikasi ini menggunakan API Gateway. Untuk informasi selengkapnya, lihat [Menambahkan IPv6 dukungan untuk VPC Anda](#).

- Titik akhir VPC untuk pribadi tunduk pada batasan APIs yang sama dengan titik akhir VPC antarmuka lainnya. Untuk informasi selengkapnya, lihat [Mengakses AWS layanan menggunakan titik akhir VPC antarmuka di Panduan](#). AWS PrivateLink Untuk informasi selengkapnya tentang penggunaan API Gateway dengan subnet bersama VPCs dan bersama, lihat [Subnet bersama](#) dalam Panduan. AWS PrivateLink

Langkah selanjutnya untuk pribadi APIs

Untuk mempelajari cara membuat API pribadi dan mengaitkan titik akhir VPC, lihat, [the section called “Buat API pribadi”](#) Untuk mengikuti tutorial di mana Anda membuat dependensi CloudFormation dan API pribadi di Konsol Manajemen AWS, lihat. [the section called “Tutorial: Membuat REST API pribadi”](#)

Buat API pribadi

Sebelum membuat API pribadi, Anda terlebih dahulu membuat titik akhir VPC untuk API Gateway. Selanjutnya Anda membuat API pribadi Anda dan melampirkan kebijakan sumber daya ke dalamnya. Secara opsional, Anda dapat mengaitkan titik akhir VPC Anda dengan API pribadi untuk menyederhanakan cara Anda menjalankan API. Terakhir, Anda menerapkan API Anda.

Prosedur berikut menjelaskan bagaimana mencapai hal ini. Anda dapat membuat REST API pribadi menggunakan Konsol Manajemen AWS, AWS CLI atau AWS SDK.

Prasyarat

Untuk mengikuti langkah-langkah ini, Anda harus memiliki VPC yang sepenuhnya dikonfigurasi. Untuk mempelajari cara membuat VPC, lihat [Membuat VPC hanya di Panduan Pengguna Amazon VPC](#). Untuk mengikuti semua langkah rekomendasi saat Anda membuat VPC, aktifkan DNS pribadi. Dengan cara ini Anda dapat menjalankan API Anda dalam VPC tanpa harus melewati Host x - apigw-api-id atau header.

Untuk mengaktifkan DNS pribadi, `enableDnsHostnames` atribut `enableDnsSupport` dan VPC Anda harus disetel ke. `true` Untuk informasi selengkapnya, lihat [Dukungan DNS di VPC Anda dan Memperbarui Dukungan DNS untuk VPC Anda](#).

Langkah 1: Buat titik akhir VPC untuk API Gateway di VPC Anda

Prosedur berikut menunjukkan cara membuat titik akhir VPC untuk API Gateway. Untuk membuat titik akhir VPC untuk API Gateway, Anda menentukan `execute-api` domain Wilayah AWS tempat Anda

membuat API pribadi. `execute-api` Domain adalah layanan komponen API Gateway untuk eksekusi API.

Saat membuat titik akhir VPC untuk API Gateway, Anda menentukan setelan DNS. Jika Anda mematikan DNS pribadi, Anda hanya dapat mengakses API menggunakan DNS publik. Untuk informasi selengkapnya, lihat [the section called “Masalah: Saya tidak dapat terhubung ke API publik saya dari titik akhir VPC API Gateway”](#).

Konsol Manajemen AWS

Untuk membuat antarmuka VPC endpoint untuk API Gateway

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di <https://console.aws.amazon.com/vpc/>
2. Di panel navigasi, di bawah Virtual private cloud, pilih Endpoints.
3. Pilih Buat titik akhir.
4. (Opsional) Untuk tag Nama, masukkan nama untuk membantu mengidentifikasi titik akhir VPC Anda.
5. Untuk kategori Layanan, pilih AWS layanan.
6. Di bawah Layanan, di bilah pencarian, masukkan **execute-api**. Kemudian, pilih titik akhir layanan API Gateway di Wilayah AWS tempat Anda akan membuat API. Nama layanan akan terlihat seperti `com.amazonaws.us-east-1.execute-api` dan Type harus Interface.
7. Untuk VPC, pilih VPC tempat Anda ingin membuat endpoint.
8. (Opsional) Untuk mematikan Aktifkan Nama DNS Pribadi, pilih Pengaturan tambahan dan kemudian hapus Aktifkan Nama DNS Pribadi.
9. Untuk Subnet, pilih Availability Zones tempat Anda membuat antarmuka jaringan endpoint. Untuk meningkatkan ketersediaan API Anda, pilih beberapa subnet.
10. Untuk grup Keamanan, pilih grup keamanan untuk dikaitkan dengan antarmuka jaringan titik akhir VPC.

Grup keamanan yang Anda pilih harus diatur untuk mengizinkan lalu lintas HTTPS masuk TCP Port 443 dari rentang IP di VPC Anda atau grup keamanan lain di VPC Anda.

11. Untuk Kebijakan, lakukan salah satu hal berikut:
 - Jika Anda belum membuat API pribadi atau tidak ingin mengonfigurasi kebijakan titik akhir VPC kustom, pilih Akses penuh.

- Jika Anda telah membuat API pribadi dan ingin mengonfigurasi kebijakan titik akhir VPC kustom, Anda dapat memasukkan kebijakan titik akhir VPC kustom. Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#).

Anda dapat memperbarui kebijakan titik akhir VPC setelah membuat titik akhir VPC. Untuk informasi selengkapnya, lihat [Memperbarui kebijakan titik akhir VPC](#).

12. Pilih Buat titik akhir.
13. Salin ID titik akhir VPC yang dihasilkan, karena Anda mungkin menggunakannya di langkah-langkah masa depan.

AWS CLI

[create-vpc-endpoint](#) Perintah berikut membuat titik akhir VPC:

```
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
```

Salin ID titik akhir VPC yang dihasilkan, karena Anda mungkin menggunakannya di langkah-langkah masa depan.

Langkah 2: Buat API pribadi

Setelah membuat titik akhir VPC, Anda membuat REST API pribadi. Prosedur berikut menunjukkan cara membuat API pribadi.

Konsol Manajemen AWS

Untuk membuat API pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Buat API.
3. Di bawah REST API, pilih Build.
4. Untuk Nama, masukkan nama.

5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Untuk jenis endpoint API, pilih Private.
7. (Opsional) Untuk titik akhir VPC IDs, masukkan ID titik akhir VPC.

Jika Anda mengaitkan ID titik akhir VPC dengan API pribadi, Anda dapat menjalankan API dari dalam VPC tanpa mengganti Host header atau meneruskan Untuk informasi selengkapnya, lihat. `x-apigw-api-id` header [the section called “\(Opsional\) Kaitkan atau lepaskan titik akhir VPC dengan API pribadi”](#)

8. Untuk jenis alamat IP, pilih Dualstack.
9. Pilih Buat API.

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk [the section called “Memulai dengan konsol REST API”](#) untuk menyiapkan metode dan integrasi untuk API ini, tetapi Anda tidak dapat menerapkan API Anda. Untuk menerapkan API Anda, ikuti langkah 3 dan lampirkan kebijakan sumber daya ke API Anda.

AWS CLI

[create-rest-api](#) Perintah berikut membuat API pribadi:

```
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" }'
```

Panggilan yang berhasil mengembalikan output yang mirip dengan berikut ini:

```
{
  "createdDate": "2017-10-13T18:41:39Z",
  "description": "Simple private PetStore API",
  "endpointConfiguration": {
    "types": [
      "PRIVATE"
    ],
    "ipAddressType": "dualstack"
  },
  "id": "0qzs2sy7bh",
  "name": "Simple PetStore (AWS CLI, Private)"
}
```

```
}
```

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk [the section called “Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI”](#) untuk menyiapkan metode dan integrasi untuk API ini, tetapi Anda tidak dapat menerapkan API Anda. Untuk menerapkan API Anda, ikuti langkah 3 dan lampirkan kebijakan sumber daya ke API Anda.

SDK JavaScript v3

Contoh berikut menunjukkan cara membuat API pribadi dengan menggunakan AWS SDK untuk JavaScript v3:

```
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)
  }
};
```

Panggilan yang berhasil mengembalikan output yang mirip dengan berikut ini:

```
{
  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'
}
```

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk [the section called “Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI”](#) untuk menyiapkan metode dan integrasi untuk API ini, tetapi Anda tidak dapat menerapkan API Anda. Untuk menerapkan API Anda, ikuti langkah 3 dan lampirkan kebijakan sumber daya ke API Anda.

Python SDK

Contoh berikut menunjukkan cara membuat API pribadi dengan menggunakan AWS SDK untuk Python:

```
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
```

Panggilan yang berhasil mengembalikan output yang mirip dengan berikut ini:

```
{\\"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\\"}
```

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk [the section called “Tutorial: Buat REST API menggunakan AWS SDKs atau AWS CLI”](#) untuk menyiapkan metode dan integrasi untuk API ini, tetapi Anda tidak dapat menerapkan API Anda. Untuk menerapkan API Anda, ikuti langkah 3 dan lampirkan kebijakan sumber daya ke API Anda.

Langkah 3: Siapkan kebijakan sumber daya untuk API pribadi

API pribadi Anda saat ini tidak dapat diakses oleh semua VPCs orang. Gunakan kebijakan sumber daya untuk memberikan akses titik akhir Anda VPCs dan VPC ke pribadi Anda. APIs Anda dapat memberikan akses ke titik akhir VPC di akun apa pun. AWS

Kebijakan sumber daya Anda harus berisi `aws:SourceVpc` atau `aws:SourceVpce` ketentuan untuk membatasi akses. Kami menyarankan Anda mengidentifikasi titik akhir spesifik VPCs dan VPC dan tidak membuat kebijakan sumber daya yang memungkinkan akses untuk semua dan titik akhir VPCs VPC.

Prosedur berikut menunjukkan cara melampirkan kebijakan sumber daya ke API Anda.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Pilih Buat kebijakan.
5. Pilih Pilih template dan kemudian pilih Sumber VPC.
6. Ganti `{{vpcID}}` (termasuk kurung kurawal) dengan ID VPC Anda.
7. Pilih Simpan perubahan.

AWS CLI

[update-rest-api](#)Perintah berikut melampirkan kebijakan sumber daya ke API yang ada:

```
aws apigateway update-rest-api \  
  --rest-api-id a1b2c3 \  
  --patch-operations op=replace,path=/  
policy,value='{"jsonEscapedPolicyDocument"}'
```

Anda mungkin juga ingin mengontrol sumber daya mana yang memiliki akses ke titik akhir VPC Anda. Untuk mengontrol sumber daya mana yang memiliki akses ke titik akhir VPC Anda, lampirkan kebijakan titik akhir ke titik akhir VPC Anda. Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#).

(Optional) Kaitkan atau lepaskan titik akhir VPC dengan API pribadi

Saat Anda mengaitkan titik akhir VPC dengan API pribadi Anda, API Gateway menghasilkan catatan DNS alias Route 53 baru. Anda dapat menggunakan catatan ini untuk memanggil pribadi Anda APIs seperti yang Anda lakukan publik Anda APIs tanpa mengganti Host header atau meneruskan header. `x-apigw-api-id`

URL dasar yang dihasilkan dalam format berikut:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (Konsol Manajemen AWS)

Anda dapat mengaitkan titik akhir VPC dengan API pribadi saat membuatnya, atau setelah dibuat. Prosedur berikut menunjukkan cara mengaitkan titik akhir VPC dengan API yang dibuat sebelumnya.

Untuk mengaitkan titik akhir VPC dengan API pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API pribadi Anda.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Edit kebijakan sumber daya Anda untuk mengizinkan panggilan dari titik akhir VPC tambahan Anda.
5. Di panel navigasi utama, pilih pengaturan API.
6. Di bagian detail API, pilih Edit.
7. Untuk titik akhir VPC IDs, pilih titik akhir VPC tambahan. IDs

8. Pilih Simpan.
9. Menerapkan ulang API Anda agar perubahan diterapkan.

Dissociate a VPC endpoint (Konsol Manajemen AWS)

Untuk memisahkan titik akhir VPC dari REST API pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API pribadi Anda.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Edit kebijakan sumber daya Anda untuk menghapus penyebutan titik akhir VPC yang ingin Anda pisahkan dari API pribadi Anda.
5. Di panel navigasi utama, pilih pengaturan API.
6. Di bagian detail API, pilih Edit.
7. Untuk titik akhir VPC IDs, pilih X untuk memisahkan titik akhir VPC.
8. Pilih Simpan.
9. Menerapkan ulang API Anda agar perubahan diterapkan.

Associate a VPC endpoint (AWS CLI)

[create-rest-api](#) Perintah berikut mengaitkan titik akhir VPC pada saat pembuatan API:

```
aws apigateway create-rest-api \
  --name Petstore \
  --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
  --region us-west-2
```

Outputnya akan terlihat seperti berikut:

```
{
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "PRIVATE"
    ],
    "vpcEndpointIds": [
```

```

        "vpce-0212a4ababd5b8c3e",
        "vpce-0393a628149c867ee"
    ]
},
"id": "u67n3ov968",
"createdDate": 1565718256,
"name": "Petstore"
}

```

[update-rest-api](#) Perintah berikut mengaitkan titik akhir VPC ke API yang sudah Anda buat:

```

aws apigateway update-rest-api \
  --rest-api-id u67n3ov968 \
  --patch-operations "op='add',path='/endpointConfiguration/vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
  --region us-west-2

```

Outputnya akan terlihat seperti berikut:

```

{
  "name": "Petstore",
  "apiKeySource": "1565718256",
  "tags": {},
  "createdDate": 1565718256,
  "endpointConfiguration": {
    "vpcEndpointIds": [
      "vpce-0212a4ababd5b8c3e",
      "vpce-0393a628149c867ee",
      "vpce-01d622316a7df47f9"
    ],
    "types": [
      "PRIVATE"
    ]
  },
  "id": "u67n3ov968"
}

```

Menerapkan ulang API Anda agar perubahan diterapkan.

Disassociate a VPC endpoint (AWS CLI)

[update-rest-api](#) Perintah berikut memisahkan titik akhir VPC dari API pribadi:


```
aws apigateway update-rest-api \  
  --rest-api-id u67n3ov968 \  
  --patch-operations "op='remove',path='/endpointConfiguration/  
vpcEndpointIds',value='vpce-0393a628149c867ee'" \  
  --region us-west-2
```

Outputnya akan terlihat seperti berikut:

```
{  
  "name": "Petstore",  
  "apiKeySource": "1565718256",  
  "tags": {},  
  "createdDate": 1565718256,  
  "endpointConfiguration": {  
    "vpcEndpointIds": [  
      "vpce-0212a4ababd5b8c3e",  
      "vpce-01d622316a7df47f9"  
    ],  
    "types": [  
      "PRIVATE"  
    ]  
  },  
  "id": "u67n3ov968"  
}
```

Menerapkan ulang API Anda agar perubahan diterapkan.

Langkah 4: Menerapkan API pribadi

Untuk menerapkan API, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung. Prosedur berikut menunjukkan cara menerapkan API pribadi Anda.

Konsol Manajemen AWS

Untuk menerapkan API pribadi

1. Pilih API Anda.
2. Pilih Deploy API.
3. Untuk Stage, pilih New stage.
4. Untuk nama Panggung, masukkan nama panggung.

5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Pilih Deploy.

AWS CLI

Perintah [create-deployment](#) berikut menerapkan API pribadi:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \  
  --stage-name test \  
  --stage-description 'Private API test stage' \  
  --description 'First deployment'
```

Memecahkan masalah API pribadi Anda

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat membuat API pribadi.

Masalah: Saya tidak dapat terhubung ke API publik saya dari titik akhir VPC API Gateway

Saat Anda membuat VPC, Anda dapat mengonfigurasi pengaturan DNS. Kami menyarankan Anda mengaktifkan DNS pribadi untuk VPC Anda. Jika Anda memilih mematikan DNS pribadi, Anda hanya dapat mengakses API Anda melalui DNS publik.

Jika Anda mengaktifkan DNS pribadi, Anda tidak dapat mengakses titik akhir default API Gateway API publik dari titik akhir VPC Anda. Anda dapat mengakses API dengan nama domain khusus.

Jika Anda membuat nama domain kustom Regional, gunakan catatan alias tipe A, jika Anda membuat nama domain kustom yang dioptimalkan tepi, tidak ada batasan untuk jenis rekaman Anda. Anda dapat mengakses publik ini APIs dengan DNS pribadi diaktifkan. Untuk informasi selengkapnya, lihat [Masalah: Saya terhubung ke API publik saya dari titik akhir VPC API Gateway](#).

Masalah: API saya kembali **{"Message": "User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:*****/****/****/"}**

Dalam kebijakan sumber daya Anda, jika Anda menetapkan Principal ke AWS prinsipal, seperti berikut ini:

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"
      ]
    }
  ]
}
```

Anda harus menggunakan AWS_IAM otorisasi untuk setiap metode di API Anda, atau API Anda mengembalikan pesan kesalahan sebelumnya. Untuk petunjuk selengkapnya tentang cara mengaktifkan AWS_IAM otorisasi untuk suatu metode, lihat [the section called “Metode”](#).

Masalah: Saya tidak tahu apakah titik akhir VPC saya dikaitkan dengan API saya

Jika Anda mengaitkan atau memisahkan titik akhir VPC dengan API pribadi, Anda perlu menerapkan ulang API Anda. Operasi pembaruan mungkin memakan waktu beberapa menit untuk diselesaikan karena propagasi DNS. Selama waktu ini, API Anda tersedia, tetapi propagasi DNS untuk DNS yang baru dibuat URLs mungkin masih dalam proses. Jika setelah beberapa menit, yang baru URLs Anda tidak diselesaikan di DNS, kami sarankan Anda menerapkan ulang API Anda.

Nama domain khusus untuk pribadi APIs di API Gateway

Anda dapat membuat nama domain khusus untuk pribadi Anda APIs. Gunakan nama domain kustom pribadi untuk memberikan URL yang lebih sederhana dan intuitif kepada pemanggil API. Dengan nama domain kustom pribadi, Anda dapat mengurangi kerumitan, mengonfigurasi langkah-langkah keamanan selama jabat tangan TLS, dan mengontrol siklus hidup sertifikat nama domain Anda

menggunakan AWS Certificate Manager (ACM). Untuk informasi selengkapnya, lihat [the section called “Mengamankan kunci pribadi sertifikat Anda untuk nama domain kustom Anda”](#).

Nama domain khusus untuk pribadi APIs tidak perlu unik di beberapa akun. Anda dapat membuat `example.private.com` di akun 111122223333 dan di akun 55555555555555, selama sertifikat ACM Anda mencakup nama domain. Untuk mengidentifikasi nama domain kustom pribadi, gunakan nama domain kustom pribadi ARN. Pengenal ini unik untuk nama domain kustom pribadi.

Saat membuat nama domain kustom pribadi di API Gateway, Anda adalah penyedia API. Anda dapat memberikan nama domain kustom pribadi Anda kepada orang lain Akun AWS menggunakan API Gateway atau AWS Resource Access Manager (AWS RAM).

Saat Anda memanggil nama domain kustom pribadi, Anda adalah konsumen API. Anda dapat menggunakan nama domain kustom pribadi dari Anda sendiri Akun AWS atau dari yang lain Akun AWS.

Saat menggunakan nama domain kustom pribadi, Anda membuat asosiasi akses nama domain antara titik akhir VPC dan nama domain kustom pribadi. Dengan asosiasi akses nama domain, konsumen API dapat memanggil nama domain kustom pribadi Anda saat diisolasi dari internet publik. Untuk informasi selengkapnya, lihat [the section called “Penyedia API dan konsumen API”](#).

Mengamankan kunci pribadi sertifikat Anda untuk nama domain kustom Anda

Saat Anda meminta SSL/TLS sertifikat menggunakan ACM untuk membuat nama domain kustom Anda secara pribadi APIs, ACM menghasilkan public/private key pair. Saat Anda mengimpor sertifikat, Anda menghasilkan key pair. Kunci publik menjadi bagian dari sertifikat. Untuk menyimpan kunci pribadi dengan aman, ACM membuat kunci lain menggunakan AWS KMS, yang disebut kunci KMS, dengan alias `aws/acm`. AWS KMS menggunakan kunci ini untuk mengenkripsi kunci pribadi sertifikat Anda. Untuk informasi selengkapnya, lihat [Perlindungan data AWS Certificate Manager di Panduan AWS Certificate Manager Pengguna](#).

API Gateway menggunakan AWS TLS Connection Manager, layanan yang hanya dapat diakses Layanan AWS, untuk mengamankan dan menggunakan kunci pribadi sertifikat Anda. Saat Anda menggunakan sertifikat ACM untuk membuat nama domain kustom API Gateway, API Gateway mengaitkan sertifikat Anda dengan AWS TLS Connection Manager. Kami melakukan ini dengan membuat hibah AWS KMS terhadap kunci AWS terkelola Anda. Hibah ini memungkinkan TLS Connection Manager digunakan AWS KMS untuk mendekripsi kunci pribadi sertifikat Anda. TLS Connection Manager menggunakan sertifikat dan kunci pribadi yang didekripsi (plaintext) untuk membuat koneksi aman (sesi SSL/TLS) dengan klien layanan API Gateway. Ketika sertifikat

diputusan dari layanan API Gateway, hibah akan dihentikan. Untuk informasi selengkapnya, lihat [Hibah](#) di Panduan AWS Key Management Service Pengembang.

Untuk informasi selengkapnya, lihat [the section called “Enkripsi diam”](#).

Pertimbangan untuk nama domain kustom pribadi

Pertimbangan berikut dapat memengaruhi penggunaan nama domain kustom pribadi Anda:

- API Gateway membutuhkan waktu sekitar 15 menit untuk menyediakan nama domain kustom pribadi Anda.
- Jika Anda memperbarui sertifikat ACM, API Gateway membutuhkan waktu sekitar 15 menit untuk menyelesaikan pembaruan. Selama waktu ini, nama domain Anda berada di UPDATING negara bagian, dan Anda masih dapat mengaksesnya.
- Untuk memanggil nama domain kustom pribadi, Anda harus membuat asosiasi akses nama domain. Setelah Anda membuat asosiasi akses nama domain, dibutuhkan sekitar 15 menit untuk siap.
- Nama domain kustom pribadi ARN berisi *account-id* dan file. *domain-name-id* Saat Anda membuat nama domain, API Gateway menggunakan format ARN.
`arn:partition:apigateway:region::/domainnames/domain-name` Saat Anda mengakses nama domain kustom pribadi, Anda menggunakan format ARN.
`arn:partition:apigateway:region:account-id:/domainnames/domain-name+domain-name-id`

Anda mungkin perlu mengubah izin IAM Anda untuk mengizinkan akses ke nama domain pribadi setelah Anda membuatnya.

- Anda tidak dapat memanggil nama domain kustom pribadi dengan nama yang sama dari titik akhir VPC yang sama. Misalnya, jika Anda ingin memanggil `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234` dan `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+xyz000`, kaitkan setiap nama domain kustom pribadi dengan titik akhir VPC yang berbeda.
- Sertifikat wildcard didukung, seperti sertifikat untuk `*.private.example.com`.
- Nama domain kustom wildcard tidak didukung.
- Hanya sertifikat RSA dengan panjang kunci 2048-bit dan sertifikat ECDSA dengan panjang kunci 256-bit dan 384-bit yang didukung.

- Anda tidak dapat mengatur jenis alamat IP untuk pribadi APIs agar hanya mengizinkan IPv4 alamat menjalankan API pribadi Anda. Hanya dualstack yang didukung. Untuk informasi selengkapnya, lihat [the section called “Jenis alamat IP untuk REST APIs di API Gateway”](#).
- Untuk mengirim lalu lintas menggunakan API pribadi, Anda dapat menggunakan semua jenis alamat IP yang didukung oleh Amazon VPC. Anda dapat mengirim dualstack dan IPv6 lalu lintas dengan mengonfigurasi pengaturan pada titik akhir VPC Anda. Anda tidak dapat memodifikasi ini menggunakan API Gateway. Untuk informasi selengkapnya, lihat [Menambahkan IPv6 dukungan untuk VPC Anda](#).
- Pemetaan jalur dasar multi-level, seperti memetakan API pribadi Anda ke `/developers/feature`, tidak didukung, tetapi Anda dapat menggunakan aturan perutean untuk membuat kondisi jalur multi-level. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).
- Anda tidak dapat menetapkan versi TLS minimum untuk nama domain kustom pribadi Anda. Semua nama domain kustom pribadi memiliki kebijakan keamanan TLS-1-2.
- Anda dapat menggunakan kebijakan titik akhir VPC untuk mengontrol akses ke nama domain kustom pribadi. Untuk informasi lebih lanjut, lihat contoh 4 dan 5 in [the section called “Menggunakan kebijakan titik akhir VPC untuk pribadi APIs”](#).
- Anda harus membuat kebijakan sumber daya terpisah untuk API pribadi Anda dan untuk nama domain kustom pribadi Anda. Untuk memanggil nama domain kustom pribadi, konsumen API memerlukan akses dari kebijakan sumber daya nama domain kustom pribadi, kebijakan sumber daya API pribadi, dan kebijakan atau otorisasi titik akhir VPC apa pun pada API pribadi.

Pertimbangan untuk menggunakan nama domain kustom pribadi dengan sumber daya API Gateway lainnya

Pertimbangan berikut dapat memengaruhi cara Anda menggunakan nama domain kustom pribadi dengan sumber daya API Gateway lainnya:

- Anda tidak dapat mengirim lalu lintas dari nama domain kustom pribadi ke API publik.
- Saat API pribadi dipetakan ke nama domain kustom pribadi, Anda tidak dapat mengubah jenis titik akhir API.
- Anda tidak dapat memigrasikan nama domain kustom publik ke nama domain kustom pribadi.
- Jika Anda memiliki titik akhir VPC yang Anda gunakan untuk mengakses nama domain kustom publik, jangan menggunakannya untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi.

Perbedaan antara nama domain kustom pribadi dan nama domain kustom publik

Berikut ini menjelaskan perbedaan antara nama domain kustom pribadi dan publik:

- Nama domain kustom pribadi tidak perlu unik di beberapa akun.
- Nama domain pribadi memiliki ID nama domain. ID ini secara unik mengidentifikasi nama domain kustom pribadi dan tidak dibuat untuk nama domain kustom publik.
- Ketika Anda menggunakan AWS CLI untuk memperbarui atau menghapus nama domain kustom pribadi Anda, Anda harus memberikan ID nama domain. Jika Anda memiliki nama domain kustom pribadi yang dipanggil `example.com` dan nama domain kustom publik dipanggil `example.com` dan Anda tidak memberikan ID nama domain, API Gateway akan memodifikasi atau menghapus nama domain kustom publik Anda.

Langkah selanjutnya untuk nama domain khusus untuk pribadi APIs

Untuk informasi tentang tugas penyedia API dan konsumen API, lihat [the section called “Penyedia API dan konsumen API”](#).

Untuk petunjuk cara membuat nama domain kustom pribadi yang dapat Anda panggil sendiri Akun AWS, lihat [the section called “Tutorial: Membuat dan memanggil nama domain khusus untuk pribadi APIs”](#).

Untuk petunjuk tentang menyediakan Akun AWS akses lain ke nama domain kustom pribadi Anda, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM”](#). Untuk petunjuk tentang mengaitkan titik akhir VPC Anda dengan nama domain kustom pribadi di Akun AWS tempat lain, lihat [the section called “Konsumen API: Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi yang dibagikan dengan Anda”](#)

Tugas penyedia API dan konsumen API untuk nama domain khusus untuk pribadi APIs

Saat membuat nama domain kustom pribadi, Anda adalah penyedia API. Saat Anda memanggil nama domain kustom pribadi, Anda adalah konsumen API. Anda dapat menggunakan nama domain kustom pribadi dari Anda sendiri Akun AWS atau dari yang lain Akun AWS.

Bagian berikut menjelaskan tugas yang diperlukan oleh penyedia API dan konsumen API untuk menggunakan nama domain kustom pribadi. Jika Anda ingin memanggil nama domain kustom pribadi sendiri Akun AWS, Anda adalah penyedia API dan konsumen API. Jika Anda ingin memanggil domain kustom pribadi di domain lain Akun AWS, tergantung pada hubungan kepercayaan antara

penyedia API dan konsumen API di AWS Organizations, AWS RAM mungkin menyelesaikan beberapa tugas untuk Anda.

Tugas penyedia API

Penyedia API membuat pribadi APIs dan memetakannya ke nama domain khusus.

Penyedia API mengelola dua kebijakan sumber daya untuk melindungi nama domain kustom pribadi mereka. Kebijakan pertama adalah untuk `execute-api` layanan dan kontrol titik akhir VPC mana yang dapat memanggil nama domain kustom pribadi Anda. Dalam konfigurasi nama domain kustom pribadi, itu disebut `policy`.

Kebijakan kedua adalah untuk layanan Manajemen Amazon API Gateway dan mengontrol titik akhir VPC mana yang lain Akun AWS dapat membentuk asosiasi akses nama domain dengan nama domain kustom pribadi Anda. Titik akhir VPC perlu membentuk asosiasi akses nama domain dengan nama domain kustom pribadi untuk memanggilnya. Dalam konfigurasi nama domain kustom pribadi, itu adalah `managementPolicy`. Anda dapat menggunakan AWS RAM atau API Gateway untuk memperbarui kebijakan ini. Jika Anda tidak berencana mengizinkan titik akhir VPC di lain Akun AWS untuk memanggil nama domain kustom Anda, Anda tidak mengedit `managementPolicy`.

Jika Anda adalah penyedia API, Anda harus melakukan hal berikut:

1. Buat API pribadi.
2. Perbarui API pribadi Anda `policy` untuk memberikan akses titik akhir VPC Anda ke API pribadi Anda.
3. Buat nama domain kustom pribadi.
4. Perbarui nama domain kustom pribadi Anda `policy` untuk memberikan akses titik akhir VPC Anda ke nama domain kustom pribadi Anda.
5. Buat pemetaan jalur dasar atau aturan perutean untuk mengirim lalu lintas dari API pribadi Anda ke nama domain kustom pribadi Anda. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

Jika Anda ingin mengizinkan konsumen API di pihak lain Akun AWS untuk mengakses nama domain kustom pribadi Anda, lakukan hal berikut:

1. `managementPolicy` Perbarui nama domain kustom pribadi Anda untuk memungkinkan konsumen API di akun lain mengaitkan titik akhir VPC mereka dengan nama domain kustom pribadi Anda. Anda dapat melakukan ini menggunakan metode berikut:

AWS RAM

Dengan AWS RAM, jika penyedia API dan konsumen API berada di organisasi yang sama menggunakan AWS Organizations, pembagian sumber daya antara penyedia dan konsumen akan diterima secara otomatis. Jika tidak, Anda harus menunggu hingga konsumen API menerima pembagian sumber daya. Kami menyarankan Anda menggunakan AWS RAM untuk membagikan nama domain kustom pribadi Anda.

API Gateway

Dengan API Gateway, hanya AWS CLI yang didukung. Anda harus memperbarui nama domain kustom pribadi Anda menggunakan operasi tambalan dan memberikan dokumen kebijakan Anda sendiri untuk `managementPolicy`.

2. `policy` Perbarui nama domain kustom pribadi Anda dan setiap pribadi yang APIs dipetakan ke sana untuk memberikan akses ke titik akhir VPC konsumen API.

Untuk petunjuk tentang cara menyediakan API Anda ke yang lain Akun AWS, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM”](#).

Tugas konsumen API

Konsumen API mengaitkan titik akhir VPC mereka dengan nama domain ARN untuk memanggil nama domain kustom pribadi. Konsumen API tidak perlu membuat API Gateway API.

Jika Anda adalah konsumen API, lakukan hal berikut:

1. Buat titik akhir VPC dengan DNS pribadi yang diaktifkan di Amazon VPC.
2. (Opsional - jika AWS RAM digunakan) Terima pembagian sumber daya domain kustom pribadi AWS RAM dalam waktu 12 jam setelah pembagian sumber daya. Jika Anda dan penyedia API berada di organisasi yang sama, pembagian sumber daya akan diterima secara otomatis.
3. Dapatkan nama domain kustom pribadi ARN. Karena URL nama domain kustom pribadi tidak unik, Anda menggunakan nama domain kustom pribadi ARN untuk membentuk asosiasi akses nama domain antara titik akhir VPC Anda dan nama domain kustom pribadi. Anda dapat menggunakan AWS RAM untuk mengambil nama domain kustom pribadi ARN.
4. Kaitkan ARN domain kustom pribadi dengan titik akhir VPC Anda di API Gateway. Ini menciptakan koneksi aman antara titik akhir VPC Anda dan nama domain kustom pribadi. Lalu lintas tidak meninggalkan jaringan Amazon.

5. Tunggu penyedia API memberikan akses titik akhir VPC Anda ke nama domain kustom pribadi dan setiap pribadi yang APIs dipetakan ke nama domain kustom pribadi. Jika Anda adalah penyedia API dan konsumen API, Anda memberikan akses pemanggilan titik akhir VPC Anda sendiri.
6. Buat Route 53 Private Hosted Zone dan catatan Route 53 untuk menyelesaikan nama domain kustom pribadi di Route 53.

Untuk petunjuk tentang cara menggunakan API di API lain Akun AWS, lihat [the section called “Konsumen API: Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi yang dibagikan dengan Anda”](#).

Tutorial: Membuat dan memanggil nama domain khusus untuk pribadi APIs

Dalam tutorial ini, Anda membuat nama domain kustom pribadi yang dapat Anda panggil dalam VPC di akun Anda sendiri. Untuk mencapai hal ini, Anda adalah penyedia API dan konsumen API. Anda memerlukan API pribadi dan titik akhir VPC yang ada untuk menyelesaikan tutorial ini. Jika Anda memiliki titik akhir VPC yang Anda gunakan untuk mengakses nama domain kustom publik, jangan menggunakannya untuk tutorial ini atau untuk membuat asosiasi akses nama domain apa pun.

Langkah 1: Buat nama domain kustom pribadi

Anda membuat nama domain kustom pribadi dengan menentukan nama domain, sertifikat ACM, dan kebijakan `execute-api` layanan untuk mengontrol titik akhir VPC mana yang dapat memanggilnya.

Konsol Manajemen AWS

Untuk membuat nama domain kustom pribadi

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih Tambahkan nama domain.
4. Untuk nama Domain, masukkan nama domain.

Sertifikat ACM Anda harus mencakup nama domain ini, tetapi nama domain tidak harus unik.

5. Pilih Pribadi.
6. Untuk mode Routing, pilih pemetaan API saja.
7. Untuk sertifikat ACM, pilih sertifikat.

8. Pilih Tambahkan nama domain.

API Gateway menyediakan nama domain dengan kebijakan deny semua sumber daya. Ini adalah kebijakan sumber daya untuk `execute-api` layanan ini. Anda perlu memperbarui kebijakan sumber daya ini untuk memberikan akses ke titik akhir VPC Anda untuk memanggil nama domain kustom pribadi Anda.

Untuk memperbarui kebijakan sumber daya Anda

1. Pilih tab Kebijakan sumber daya, lalu pilih Edit kebijakan sumber daya.
2. Masukkan kebijakan sumber daya berikut di editor kode. Ganti titik akhir VPC `vpce-abcd1234efg` dengan ID titik akhir VPC Anda sendiri.

```
{
  "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. Pilih Simpan perubahan.

AWS CLI

Saat Anda membuat nama domain kustom pribadi menggunakan AWS CLI, Anda memberikan kebijakan sumber daya untuk `execute-api` layanan untuk memberikan akses ke titik akhir VPC untuk memanggil nama domain kustom pribadi Anda, menggunakan parameter. `--policy file://policy.json` Anda dapat mengubah kebijakan ini nanti.

Untuk contoh ini, Anda akan melampirkan kebijakan sumber daya berikut sebagai parameter `policy` dengan memuat dari file. Salin dan simpan file ini sebagai `policy.json`. Kebijakan ini hanya mengizinkan lalu lintas masuk ke nama domain kustom pribadi dari titik akhir `vpce-abcd1234efg` VPC:

```
{
  "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"
        }
      }
    }
  ]
}
```

[create-domain-name](#) Perintah berikut membuat nama domain kustom pribadi:

```
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
```

Outputnya akan seperti berikut ini.

```
{  
  "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": "..."  
}
```

Langkah 2: Buat pemetaan jalur dasar untuk memetakan API pribadi Anda ke nama domain kustom pribadi Anda

Setelah Anda membuat nama domain kustom pribadi Anda, Anda memetakan API pribadi ke sana. Pemetaan jalur dasar membuat API dapat diakses melalui kombinasi nama domain kustom pribadi dan jalur dasar terkait. Kami menyarankan Anda menggunakan satu nama domain kustom pribadi sebagai nama host dari beberapa pribadi APIs.

Semua penyedia API perlu membuat pemetaan jalur dasar, bahkan jika Anda tidak berencana menjalankan API Anda sendiri. Anda juga perlu memberikan akses untuk titik akhir VPC untuk memanggil pribadi apa pun APIs yang Anda petakan ke nama domain kustom pribadi Anda.

Konsol Manajemen AWS

Untuk membuat pemetaan jalur dasar

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi.
4. Pada tab pemetaan API, pilih Konfigurasi pemetaan.
5. Pilih Tambahkan pemetaan baru.
6. Masukkan API, Stage, dan secara opsional Path.
7. Pilih Simpan.

AWS CLI

[create-base-path-mapping](#) Perintah berikut membuat pemetaan antara API pribadi dan nama domain kustom pribadi:

```
aws apigateway create-base-path-mapping \
  --domain-name-id abcd1234 \
  --domain-name 'private.example.com' \
  --rest-api-id a1b2c3 \
  --stage prod \
  --base-path v1
```

Outputnya akan terlihat seperti berikut ini.

```
{
  "basePath": "v1",
  "restApiId": "a1b2c3",
  "stage": "prod"
}
```

Untuk fleksibilitas lebih lanjut tentang cara Anda merutekan lalu lintas ke Anda APIs, Anda dapat mengubah mode perutean ke `ROUTING_RULE_ONLY` atau `ROUTING_RULE_THEN_API_MAPPING` dan membuat aturan perutean. Untuk informasi selengkapnya, lihat [the section called “Kirim lalu lintas ke Anda APIs melalui nama domain kustom Anda di API Gateway”](#).

Note

Jika Anda ingin orang lain Akun AWS memanggil nama domain kustom pribadi Anda, setelah Anda menyelesaikan tutorial ini, ikuti langkah-langkahnya. [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM”](#)

Langkah 3: Buat asosiasi akses nama domain antara nama domain khusus Anda dan titik akhir VPC

Selanjutnya, Anda membuat asosiasi akses nama domain antara nama domain kustom pribadi Anda dan titik akhir VPC Anda. Titik akhir VPC Anda menggunakan asosiasi akses nama domain untuk memanggil nama domain kustom pribadi Anda saat diisolasi dari internet publik.

Konsol Manajemen AWS

Untuk membuat asosiasi akses nama domain

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi.
4. Di tab Berbagi sumber daya, untuk asosiasi akses nama domain, pilih Buat asosiasi akses nama domain.
5. Untuk nama Domain ARN, pilih nama domain Anda.
6. Untuk ID titik akhir VPC, pilih ID titik akhir VPC yang Anda berikan akses pada langkah 1.
7. Pilih Asosiasi akses nama domain.

Anda juga dapat membuat asosiasi akses nama domain menggunakan halaman Asosiasi akses nama domain konsol.

AWS CLI

`create-domain-name-access-association` Perintah berikut membuat asosiasi akses nama domain antara nama domain kustom pribadi Anda dan titik akhir VPC Anda.

```
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
```

Outputnya akan terlihat seperti berikut ini.

```
{
  "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"
}
```

Setelah Anda membuat asosiasi akses nama domain Anda, dibutuhkan sekitar 15 menit untuk siap. Sambil menunggu, Anda dapat melanjutkan dengan langkah-langkah berikut.

Langkah 4: Buat zona yang dihosting Route 53

Setelah memperbarui kebijakan sumber daya dan mengaitkan nama domain kustom pribadi Anda dengan titik akhir VPC, Anda membuat zona host pribadi di Route 53 untuk menyelesaikan nama domain kustom Anda. Zona yang dihosting adalah wadah yang menyimpan informasi tentang bagaimana Anda ingin merutekan lalu lintas untuk domain dalam satu atau lebih VPCs tanpa mengekspos sumber daya Anda ke internet. Untuk informasi selengkapnya, lihat [Menggunakan zona yang di-hosting pribadi](#).

Konsol Manajemen AWS

Untuk menggunakannya Konsol Manajemen AWS, lihat [Membuat zona yang dihosting pribadi](#) di Panduan Pengembang Amazon Route 53.

Untuk Nama, gunakan nama domain kustom pribadi Anda. Untuk ID VPC, gunakan VPC yang berisi titik akhir VPC yang Anda gunakan pada langkah sebelumnya.

AWS CLI

[create-hosted-zone](#) Perintah berikut membuat zona host pribadi:

```
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
```

Output berisi ID zona yang dihosting. Anda menggunakan ID zona yang dihosting dalam langkah-langkah berikut.

Langkah 5: Buat catatan DNS Route 53

Setelah membuat zona yang dihosting, Anda membuat catatan untuk menyelesaikan nama domain kustom pribadi Anda. Anda menggunakan ID zona yang dihosting yang Anda buat di langkah sebelumnya. Dalam contoh ini, Anda membuat tipe catatan A. Jika Anda menggunakan IPv6 untuk titik akhir VPC Anda, buat jenis catatan AAAA. Jika Anda menggunakan dualstack untuk titik akhir VPC Anda, buat AAAA dan tipe catatan A.

Konsol Manajemen AWS

Untuk menggunakannya Konsol Manajemen AWS, lihat [Merutekan lalu lintas ke API Amazon API Gateway menggunakan nama domain Anda](#).

Gunakan Quick create dan aktifkan Alias. Untuk titik akhir, gunakan nama DNS titik akhir VPC.

AWS CLI

Untuk mengonfigurasi catatan DNS Anda untuk memetakan nama domain kustom pribadi ke nama hostnya dari ID zona host yang diberikan, Anda membuat file JSON yang berisi konfigurasi untuk menyiapkan catatan DNS untuk nama domain pribadi.

Berikut ini `setup-dns-record.json` menunjukkan cara membuat A catatan DNS untuk memetakan nama domain kustom pribadi ke nama host pribadinya. Anda memberikan ID DNS VPC Anda, dan ID zona yang dihosting yang Anda buat pada langkah sebelumnya. `DNSName`

```
{
  "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
        }
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

[change-resource-record-sets](#) Perintah berikut membuat catatan DNS untuk nama domain kustom pribadi Anda:

```
aws route53 change-resource-record-sets \  
  --hosted-zone-id ZABCDEFGHI1234 \  
  --change-batch file://path/to/your/setup-dns-record.json
```

Ganti `hosted-zone-id` dengan ID Zona Dihosting Route 53 dari catatan DNS yang ditetapkan di akun Anda. Nilai `change-batch` parameter menunjuk ke file JSON.

Jika Anda tidak berencana untuk memanggil nama domain kustom pribadi Anda sendiri, setelah Anda mengonfirmasi nama domain kustom pribadi Anda berfungsi, Anda dapat menghapus sumber daya ini.

Langkah 6: Panggil nama domain kustom pribadi Anda

Anda sekarang dapat memanggil nama domain kustom pribadi Anda sendiri Akun AWS. Di VPC Anda, gunakan perintah `curl` berikut untuk mengakses nama domain kustom pribadi Anda.

```
curl https://private.example.com/v1
```

Untuk informasi selengkapnya tentang cara lain untuk menjalankan API pribadi Anda, lihat [the section called “Memanggil API pribadi menggunakan nama domain khusus”](#).

Langkah 7: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus hubungan antara titik akhir VPC Anda dan nama domain kustom pribadi Anda, lalu hapus nama domain kustom pribadi Anda.

Konsol Manajemen AWS

Untuk menghapus asosiasi akses nama domain

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Asosiasi akses nama domain.

3. Pilih asosiasi akses nama domain Anda, lalu pilih Hapus.
4. Konfirmasikan pilihan Anda, lalu pilih Hapus.

Setelah menghapus asosiasi akses nama domain, Anda dapat menghapus nama domain kustom pribadi Anda.

Untuk menghapus nama domain kustom pribadi Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi Anda.
4. Pilih Hapus.
5. Konfirmasikan pilihan Anda, lalu pilih Hapus.

Jika perlu, Anda juga dapat menghapus titik akhir VPC Anda. Untuk informasi selengkapnya, lihat [Menghapus titik akhir antarmuka](#).

AWS CLI

Untuk membersihkan

1. `delete-access-association` Perintah berikut menghapus asosiasi akses nama domain:

```
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. `delete-domain-name` Perintah berikut menghapus nama domain kustom pribadi Anda. Perintah ini juga menghapus semua pemetaan jalur dasar.

```
aws apigateway delete-domain-name \
  --domain-name test.private.com \
  --domain-name-id abcd1234
```

Jika perlu, Anda juga dapat menghapus titik akhir VPC Anda. Untuk informasi selengkapnya, lihat [Menghapus titik akhir antarmuka](#).

Praktik terbaik

Kami menyarankan Anda menggunakan praktik terbaik berikut saat membuat nama domain kustom pribadi Anda:

- Gunakan pemetaan jalur dasar atau aturan perutean untuk mengirim lalu lintas dari satu nama domain kustom pribadi ke beberapa pribadi. APIs
- Jika titik akhir VPC tidak lagi memerlukan akses ke nama domain kustom pribadi, hapus asosiasi tersebut. Selain itu, hapus titik akhir VPC dari untuk `execute-api` layanan `policy` untuk domain kustom pribadi.
- Konfigurasi setidaknya dua Availability Zone per titik akhir VPC.
- Nonaktifkan titik akhir default. Kami menyarankan Anda menonaktifkan titik akhir default untuk memungkinkan konsumen API Anda hanya memanggil API Anda dari nama domain kustom. Untuk informasi selengkapnya, lihat [the section called “Nonaktifkan titik akhir default”](#).
- Kami menyarankan Anda menyediakan zona host pribadi Route 53 dan catatan tipe-A saat Anda menyiapkan nama domain kustom pribadi Anda. Jika Anda tidak berencana untuk memanggil nama domain kustom pribadi Anda sendiri, Anda dapat menghapus sumber daya ini nanti.

Bekerja dengan nama domain kustom pribadi lintas akun

Bagian ini menjelaskan cara bekerja dengan nama domain kustom pribadi lintas akun. Anda dapat memberikan nama domain kustom pribadi ke yang lain Akun AWS dan menggunakan yang lain Akun AWS untuk memanggil nama domain kustom pribadi.

Anda dapat membagikan nama domain kustom pribadi Anda ke yang lain Akun AWS menggunakan AWS Resource Access Manager atau API Gateway. AWS Resource Access Manager (AWS RAM) membantu Anda berbagi sumber daya dengan aman di seluruh Akun AWS dan di dalam organisasi atau unit organisasi Anda (OUs). Untuk informasi lebih lanjut, lihat [Apa itu AWS Resource Access Manager](#).

Untuk petunjuk tentang cara membagikan nama domain kustom pribadi dengan Akun AWS penggunaan lain AWS RAM, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM”](#).

Untuk petunjuk tentang cara membagikan nama domain kustom pribadi dengan yang lain Akun AWS menggunakan API Gateway, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan API Gateway AWS CLI”](#).

Untuk petunjuk tentang cara menggunakan nama domain kustom pribadi di tempat lain Akun AWS, lihat [the section called “Konsumen API: Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi yang dibagikan dengan Anda”](#).

Praktik terbaik untuk bekerja dengan nama domain kustom pribadi lintas akun

Kami merekomendasikan praktik terbaik berikut untuk bekerja dengan nama domain kustom pribadi lintas akun:

- Gunakan AWS RAM untuk membagikan nama domain kustom pribadi Anda. Bila Anda menggunakan AWS RAM, Anda dapat mengurangi overhead operasional dan Anda tidak perlu membuat `managementPolicy` untuk layanan Amazon API Gateway Management.
- Gunakan `resource-owner` parameter saat Anda mencantumkan nama domain kustom pribadi atau asosiasi akses nama domain. Gunakan `resource-owner` parameter untuk hanya mencantumkan sumber daya yang dimiliki oleh Anda atau orang lain Akun AWS.

Contoh berikut menunjukkan cara mendapatkan semua asosiasi akses nama domain yang Anda miliki:

```
aws apigateway get-domain-name-access-associations --resource-owner SELF
```

Gunakan `--resource-owner OTHER_ACCOUNTS` untuk mencantumkan semua asosiasi akses nama domain yang telah dibentuk akun lain dengan nama domain kustom pribadi Anda.

Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM

Anda dapat memberikan konsumen API di Akun AWS akses lain ke nama domain kustom pribadi Anda. Di bagian ini, Anda mempelajari cara membagikan nama domain kustom pribadi Anda menggunakan AWS RAM dan cara mengontrol akses ke nama domain kustom pribadi Anda.

Pertimbangan untuk berbagi nama domain kustom pribadi Anda

Pertimbangan berikut dapat memengaruhi cara Anda memberikan akses ke nama domain kustom pribadi Anda menggunakan AWS RAM. Untuk mempelajari cara membagikan nama domain kustom pribadi Anda tanpa menggunakan AWS RAM, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan API Gateway AWS CLI”](#).

- Nama domain kustom pribadi dibagikan di Wilayah AWS level tersebut. Baik nama domain kustom pribadi dan titik akhir VPC harus sama. Wilayah AWS

- Anda dapat menggunakan satu pembagian sumber daya dengan beberapa prinsipal, dan setelah Anda membuat pembagian sumber daya, Anda dapat menambahkan lebih banyak prinsipal ke dalamnya. Kami menyarankan bahwa jika memungkinkan, Anda menggunakan kembali bagian sumber daya Anda.
- Anda selalu perlu memberikan akses titik akhir VPC konsumen API untuk memanggil nama domain kustom pribadi Anda dan setiap pribadi APIs yang dipetakan ke sana.
- Jika konsumen API dan penyedia API berada di organisasi yang sama menggunakan AWS Organizations, pembagian sumber daya akan diterima secara otomatis. Anda masih perlu membuat pembagian sumber daya menggunakan AWS RAM.
- Jika konsumen API dan penyedia API berada di organisasi yang sama menggunakan AWS Organizations dan berbagi sumber daya dalam organisasi Anda diaktifkan, prinsip apa pun di organisasi yang Anda bagikan akan secara otomatis diberikan akses ke pembagian sumber daya. Tidak perlu undangan dan Anda dapat melewati pembagian sumber daya.
- Jika konsumen API tidak menerima pembagian sumber daya dalam waktu 12 jam, penyedia API harus membagikan sumber daya tersebut lagi.
- Setelah Anda membuat pembagian sumber daya, AWS RAM update managementPolicy untuk layanan Manajemen Amazon API Gateway untuk nama domain kustom pribadi Anda untuk mencegah akses ke prinsipal tanpa akses eksplisit. allow Untuk informasi selengkapnya, lihat [Menentukan apakah permintaan diizinkan atau ditolak dalam akun](#) di Panduan Pengguna IAM.

Yang diperbarui managementPolicy akan terlihat seperti berikut:

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 telah mencegah prinsipal tanpa akses eksplisit untuk membuat asosiasi allow akses dengan nama domain kustom pribadi Anda, dengan menambahkan yang berikut:

```

"StringNotEquals": {
  "aws:PrincipalAccount": "111122223333"
}

```

Anda masih dapat menggunakan prinsipal di Akun AWS siapa yang membuat nama domain kustom pribadi untuk membuat asosiasi akses nama domain.

Izinkan akun lain untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi Anda

Pertama, Anda memberikan akses ke yang lain Akun AWS untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi Anda.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, lihat [Membuat bagian sumber daya AWS RAM di Panduan AWS RAM Pengguna](#).

Untuk Pilih jenis sumber daya, pilih Domain Kustom Pribadi API Gateway.

AWS CLI

Berikut ini [create-resource-share](#) membuat pembagian sumber daya untuk nama domain kustom pribadi Anda. Diperlukan beberapa menit untuk menyelesaikan sumber daya dan asosiasi utama. Untuk kepala sekolah, berikan ID akun atau ID Organizations, seperti. `arn:aws:organizations::123456789012:organization/o-1234abcd` Anda dapat memberikan beberapa prinsip untuk berbagi sumber daya Anda.

```
aws ram create-resource-share \
```

```
--region us-west-2 \  
--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
```

Setelah Anda memberikan akses ke akun lain Akun AWS, konsumen API di akun tersebut harus membuat asosiasi akses nama domain antara titik akhir VPC mereka dengan nama domain kustom pribadi Anda. Anda tidak dapat membuat asosiasi akses nama domain untuk mereka. Untuk informasi selengkapnya, lihat [the section called “Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi bersama”](#).

Izinkan akun lain untuk memanggil nama domain kustom pribadi Anda

Selanjutnya, Anda memberikan akses ke titik akhir VPC konsumen API untuk memanggil nama domain kustom pribadi Anda dan setiap pribadi APIs yang dipetakan ke sana.

Konsol Manajemen AWS

Untuk mengizinkan titik akhir VPC di akun lain untuk memanggil nama domain kustom pribadi Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi yang Anda bagikan dengan orang lain Akun AWS.
4. Pada tab Kebijakan sumber daya, pilih Edit kebijakan sumber daya.
5. Tambahkan ID titik akhir VPC konsumen API ke kebijakan sumber daya Anda.

Anda dapat menemukan ID titik akhir VPC konsumen API di bagian Asosiasi akses nama domain pada tab Berbagi sumber daya di halaman Detail domain nama domain kustom pribadi Anda.

6. Pilih Simpan perubahan.

AWS CLI

Berikut ini policy untuk `execute-api` layanan ini memungkinkan lalu lintas masuk ke nama domain kustom pribadi dari titik akhir VPC *vpce-abcd1234efg* dan *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"
          ]
        }
      }
    }
  ]
}
```

[update-domain-name](#) Perintah berikut menggunakan operasi patch untuk memperbarui policy untuk nama domain kustom pribadi:

```
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
    \"]}]}"
```

Penyedia API: Berhenti membagikan nama domain khusus pribadi menggunakan AWS RAM

Untuk berhenti membagikan nama domain kustom pribadi Anda, pertama-tama Anda menghentikan konsumen API dari membuat lebih banyak asosiasi akses nama domain dengan memisahkan pembagian sumber daya. Kemudian, Anda menolak asosiasi akses nama domain dan menghapus titik akhir VPC konsumen API dari layanan policy Anda. `execute-api` Konsumen API kemudian dapat menghapus asosiasi akses nama domain mereka.

Berhenti membagikan nama domain kustom pribadi Anda

Pertama, Anda menghentikan pembagian sumber daya menggunakan AWS RAM.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, lihat [Memperbarui bagian sumber daya di AWS RAM](#).

AWS CLI

Berikut ini [disassociate-resource-share](#) memisahkan pembagian sumber daya untuk nama domain kustom pribadi Anda.

```
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
```

Tolak asosiasi akses nama domain

Setelah berhenti membagikan sumber daya AWS RAM, Anda menolak asosiasi akses nama domain antara titik akhir VPC di akun lain dan nama domain kustom pribadi Anda.

 Note

Anda tidak dapat menolak asosiasi akses nama domain di akun Anda sendiri. Untuk menghentikan berbagi sumber daya, hapus asosiasi akses nama domain. Untuk informasi selengkapnya, lihat [Menghapus asosiasi akses nama domain](#).

Ketika Anda menolak asosiasi akses nama domain dengan titik akhir VPC, jika konsumen API mencoba memanggil nama domain kustom pribadi Anda, API Gateway menolak panggilan dan mengembalikan kode status. 403

Konsol Manajemen AWS

Untuk menolak asosiasi akses nama domain

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi yang Anda bagikan dengan orang lain Akun AWS.
4. Pada Berbagi sumber daya, pilih asosiasi akses nama domain yang ingin Anda tolak.
5. Pilih Tolak asosiasi.
6. Konfirmasikan pilihan Anda, lalu pilih Tolak.

AWS CLI

`reject-domain-name-access-association` Perintah berikut menolak asosiasi akses nama domain antara titik akhir VPC dan nama domain kustom pribadi Anda:

```
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-abcd1234efg \
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

Tolak akses penyedia API untuk memanggil nama domain kustom pribadi Anda

Setelah menolak asosiasi akses nama domain, Anda menghapus titik akhir VPC konsumen API dari layanan policy Anda. `execute-api`

Konsol Manajemen AWS

Untuk menghapus titik akhir VPC konsumen API dari kebijakan sumber daya Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Nama domain khusus.
3. Pilih nama domain kustom pribadi yang Anda bagikan dengan orang lain Akun AWS.
4. Pada tab Kebijakan sumber daya, pilih Edit.
5. Hapus titik akhir VPC dari kebijakan.
6. Pilih Simpan perubahan.

AWS CLI

[update-domain-name](#) Perintah berikut menggunakan operasi patch untuk memperbarui `execute-api` layanan policy untuk nama domain kustom pribadi. Ini baru policy menghapus ID titik akhir VPC tambahan yang ditambahkan di: [the section called “Izinkan akun lain untuk memanggil nama domain kustom pribadi Anda”](#)

```
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"
          }
        }
      }
    ]
  }'
```

Konsumen API kemudian harus menghapus asosiasi akses nama domain. Anda tidak dapat menghapusnya untuk mereka. Untuk informasi selengkapnya, lihat [the section called “Konsumen API: Hapus asosiasi akses nama domain Anda dengan nama domain kustom pribadi”](#).

Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan API Gateway AWS CLI

Anda dapat membagikan nama domain kustom pribadi menggunakan API Gateway AWS CLI, tetapi kami menyarankan Anda menggunakannya AWS RAM untuk mengurangi overhead operasional Anda. Untuk petunjuk tentang cara menggunakan AWS RAM untuk membagikan nama domain kustom pribadi Anda, lihat [the section called “Penyedia API: Bagikan nama domain kustom pribadi Anda menggunakan AWS RAM”](#).

Untuk membagikan nama domain kustom pribadi menggunakan API Gateway AWS CLI, Anda memberikan Akun AWS akses lain untuk membuat asosiasi akses nama domain dan memanggil nama domain kustom pribadi Anda. Anda melakukannya dengan memperbarui managementPolicy untuk layanan API Gateway Management dan policy untuk execute-api layanan untuk nama domain kustom pribadi Anda. Anda juga perlu memberikan akses untuk titik akhir VPC konsumen API dalam kebijakan sumber daya untuk setiap pribadi yang APIs dipetakan ke nama domain kustom pribadi Anda.

Konsumen API masih perlu membuat asosiasi akses nama domain di akun mereka sendiri antara titik akhir VPC mereka dan nama domain kustom pribadi Anda. Anda tidak dapat melakukan ini untuk mereka.

Berikan akses ke nama domain kustom pribadi Anda

Untuk memberikan akses ke nama domain kustom pribadi Anda

1. Untuk memperbarui layanan Manajemen Gateway API, Anda membuat file JSON yang berisi operasi tambalan untuk memperbarui kebijakan. managementPolicy Berikut ini patch-managementPolicy.json menggantikan kebijakan saat ini managementPolicy dengan contoh kebijakan yang memberikan Akun AWS 111122223333 dan 444455556666 akses untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi. private.example.com

```
[{
  "op": "replace",
  "path": "/managementPolicy",
  "value": "{\n\"Version\": \"2012-10-17\",          \"Statement\": [{\n\"Effect\n\": \"Allow\", \"Principal\": {\n\"AWS\": [\n\"arn:aws:iam::111122223333:root\n\", \"arn:aws:iam::444455556666:root\"]\n}], \"Action\":\n\"apigateway:CreateAccessAssociation\", \"Resource\": \"arn:aws:apigateway:us-\nwest-2:111122223333:/domainnames/private.example.com+abcd1234\" } ] }"
```

[update-domain-name](#) Perintah berikut memperbarui managementPolicy penggunaan patch-managementPolicy.json.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-managementPolicy.json
```

Setelah Anda memberikan akses, Anda perlu memberi tahu konsumen API bahwa mereka dapat membentuk asosiasi akses nama domain. Jika Anda menggunakan AWS RAM, AWS RAM akan melakukan langkah ini untuk Anda.

2. Untuk memperbarui execute-api layanan, Anda membuat file JSON yang berisi operasi tambalan untuk memperbarui kebijakan. policy Berikut ini patch-policy.json menggantikan kebijakan saat ini policy dengan contoh kebijakan yang memberikan dua titik akhir VPC untuk memanggil nama domain kustom pribadi. private.example.com

```
[{
  "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\"]}}}]}"
}]
```

Gunakan [update-domain-name](#) perintah berikut untuk memperbarui policy penggunaan patch-policy.json.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-policy.json
```

Tolak akses ke nama domain kustom pribadi Anda

Untuk berhenti membagikan nama domain kustom pribadi Anda, Anda harus menolak asosiasi akses nama domain antara nama domain kustom pribadi Anda dan titik akhir VPC konsumen API.

Untuk menolak akses ke nama domain kustom pribadi Anda

1. `reject-domain-name-access-association` Perintah berikut menolak asosiasi akses nama domain.

```
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-abcd1234 \
  --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

2. Ubah `patch-managementPolicy.json` untuk menghapus akses untuk akun penyedia API untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi Anda. Berikut ini `patch-managementPolicy.json` menghapus satu akun dari `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\"} ]}"
}]
```

[update-domain-name](#) Perintah berikut untuk memperbarui `managementPolicy` penggunaan `patch-managementPolicy.json`.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-managementPolicy.json
```

3. Ubah `patch-policy.json` untuk menghapus akses untuk titik akhir VPC penyedia API untuk memanggil nama domain kustom pribadi Anda. Berikut ini `patch-policy.json` menghapus ID titik akhir VPC dari `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\"}}}}]\""}
]}
```

[update-domain-name](#) Perintah berikut memperbarui policy penggunaan `patch-policy.json`.

```
aws apigateway update-domain-name \
  --domain-name private.example.com \
  --domain-name-id abcd1234 \
  --patch-operations file://patch-policy.json
```

Contoh kebijakan yang digunakan dalam prosedur ini

Bagian berikut menunjukkan contoh kebijakan yang digunakan dalam prosedur sebelumnya.

Contoh kebijakan berikut adalah untuk layanan Amazon API Gateway Management.

managementPolicy Kebijakan ini memberikan Akun AWS 111122223333 dan 444455556666 akses untuk membuat asosiasi akses nama domain dengan nama domain kustom pribadi. `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"
}
]
}

```

Contoh kebijakan berikut adalah kebijakan policy untuk execute-api layanan. Kebijakan ini memberikan vpce-abcd1234 titik akhir VPC vpce-xyzz0000 dan akses untuk memanggil nama domain kustom pribadi.

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"
          ]
        }
      }
    }
  ]
}

```

```
}
```

Konsumen API: Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi yang dibagikan dengan Anda

Prosedur berikut menunjukkan cara menggunakan nama domain pribadi di tempat lain Akun AWS. Bergantung pada hubungan kepercayaan Anda dengan penyedia API, AWS RAM mungkin menyelesaikan beberapa tugas untuk Anda.

Jika Anda berbeda Akun AWS dari nama domain kustom pribadi, Anda hanya dapat mengaitkan titik akhir VPC Anda dengan nama domain kustom pribadi dan memanggilnya. Anda tidak dapat melihat `policy` atau parameter lain dari nama domain kustom pribadi.

Prasyarat

Prasyarat berikut diperlukan untuk menggunakan nama domain kustom pribadi di tempat lain: Akun AWS

- VPC dan titik akhir VPC untuk layanan ini. `execute-api` VPC Anda harus memiliki `enableDnsHostnames` dan `enableDnsSupport` disetel ke `true`
- Kami menyarankan Anda mengonfigurasi setidaknya dua Availability Zone per titik akhir VPC.

(Opsional) Terima berbagi sumber daya domain kustom pribadi

Jika penyedia API Anda digunakan AWS RAM untuk membuat pembagian sumber daya, Anda memiliki waktu 12 jam untuk menerimanya. Jika Anda berada di organisasi yang AWS Organizations sama menggunakan penyedia API, pembagian akan diterima secara otomatis. Jika Anda berada di organisasi yang mengaktifkan sumber daya bersama otomatis, sumber daya akan dibagikan secara otomatis dengan Anda.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, lihat [Menerima dan menolak undangan berbagi sumber daya](#) di Panduan Pengguna AWS RAM

AWS CLI

Untuk menemukan semua sumber daya yang dibagikan dengan Anda, gunakan [get-resource-share-invitations](#) perintah berikut:

```
aws ram get-resource-share-invitations \  
  --region us-west-2
```

Gunakan ARN berbagi sumber daya yang dihasilkan untuk menerima undangan berbagi sumber daya. [accept-resource-share-invitation](#) Perintah berikut menerima pembagian sumber daya.

```
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
```

Kaitkan titik akhir VPC Anda dengan nama domain kustom pribadi bersama

Karena nama domain kustom pribadi tidak unik, Anda mengaitkan titik akhir VPC Anda dengan nama domain khusus ARN yang unik. Setelah Anda membuat asosiasi akses nama domain Anda, dibutuhkan waktu hingga 15 menit agar titik akhir VPC Anda berhasil memanggil nama domain kustom pribadi Anda. Jika Anda memiliki titik akhir VPC yang Anda gunakan untuk mengakses nama domain kustom publik, jangan menggunakannya untuk membuat asosiasi akses nama domain apa pun.

Konsol Manajemen AWS

Untuk mengaitkan titik akhir VPC Anda dengan nama domain kustom pribadi bersama

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Asosiasi akses nama domain.
3. Pilih Buat asosiasi akses nama domain.
4. Untuk ARN nama Domain, pilih nama domain ARN yang dibagikan penyedia API dengan Anda.

Nama domain ARN mungkin tidak muncul di daftar dropdown. Anda dapat menggunakan AWS RAM konsol untuk melihat nama domain yang dibagikan dengan Anda dan kemudian menyalin nama domain ARN dan memasukkannya ke bidang ini.

5. Untuk ID titik akhir VPC, pilih ID titik akhir VPC yang ingin Anda bentuk asosiasi akses nama domain.
6. Pilih Buat asosiasi akses nama domain.

AWS CLI

Karena nama domain kustom pribadi tidak unik, Anda mengaitkan titik akhir VPC Anda dengan nama domain khusus ARN yang unik. Untuk menemukan nama domain ARN, gunakan salah satu perintah berikut.

1. AWS RAM

Perintah [list-resources](#) berikut mencantumkan sumber daya yang dibagikan dengan Anda. Penyedia API harus telah menggunakan AWS RAM untuk berbagi domain kustom pribadi mereka dengan Anda untuk menggunakan perintah ini.

```
aws ram list-resources \
  --resource-owner OTHER-ACCOUNTS \
  --region us-west-2
  --resource-type apigateway:Domainnames
```

API Gateway

`get-domain-names` Perintah berikut mencantumkan semua nama domain kustom pribadi yang dimiliki oleh orang lain Akun AWS yang dapat Anda bentuk asosiasi akses nama domain dengan.

```
aws apigateway get-domain-names \
  --resource-owner OTHER_ACCOUNTS \
  --region us-west-2
```

- Setelah Anda mengambil ARN, gunakan API Gateway untuk membuat asosiasi akses nama domain antara titik akhir VPC Anda dan nama domain kustom pribadi bersama. Gunakan perintah `create-domain-name-access-association` berikut:

```
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"
```

Outputnya akan terlihat seperti berikut ini.

```
{
```

```

    "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"
}

```

Setelah Anda mengaitkan titik akhir VPC Anda dengan nama domain kustom pribadi, konfirmasi bahwa penyedia API Anda telah memperbarui kebijakan nama domain kustom pribadi mereka untuk mengizinkan titik akhir VPC Anda memanggil nama domain mereka. Untuk informasi selengkapnya, lihat [the section called “Izinkan akun lain untuk memanggil nama domain kustom pribadi Anda”](#).

Buat zona yang dihosting Route 53

Untuk menyelesaikan nama domain kustom pribadi, Anda perlu membuat zona host pribadi Route 53. Zona yang dihosting adalah wadah yang menyimpan informasi tentang bagaimana Anda ingin merutekan lalu lintas untuk domain dalam satu atau lebih VPCs tanpa mengekspos sumber daya Anda ke internet. Untuk informasi selengkapnya, lihat [Menggunakan zona yang di-hosting pribadi](#).

Konsol Manajemen AWS

Untuk menggunakannya Konsol Manajemen AWS, lihat [Membuat zona yang dihosting pribadi](#) di Panduan Pengembang Amazon Route 53.

Untuk Nama, gunakan nama domain kustom pribadi. Untuk ID VPC, gunakan VPC yang berisi titik akhir VPC yang Anda gunakan untuk asosiasi akses nama domain Anda.

AWS CLI

[create-hosted-zone](#) Perintah berikut membuat zona host pribadi:

```

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

```

Output berisi ID zona yang dihosting. Anda menggunakan ID zona yang dihosting dalam langkah-langkah berikut.

Membuat catatan DNS Route 53

Setelah membuat zona yang dihosting, Anda membuat catatan untuk menyelesaikan domain kustom pribadi. Dalam contoh ini, Anda membuat tipe catatan A. Jika Anda menggunakan IPv6 untuk titik akhir VPC Anda, buat jenis catatan AAAA. Jika Anda menggunakan dualstack untuk titik akhir VPC Anda, buat AAAA dan tipe catatan A.

Konsol Manajemen AWS

Untuk menggunakannya Konsol Manajemen AWS, lihat [Merutekan lalu lintas ke API Amazon API Gateway menggunakan nama domain Anda](#).

Gunakan Quick create dan aktifkan Alias. Untuk titik akhir, gunakan nama DNS titik akhir VPC.

AWS CLI

Untuk mengonfigurasi catatan DNS Anda untuk memetakan nama domain kustom pribadi ke nama hostnya dari ID zona host yang diberikan, pertama-tama buat file JSON yang berisi konfigurasi untuk menyiapkan catatan DNS untuk nama domain pribadi.

Berikut ini `setup-dns-record.json` menunjukkan cara membuat A catatan DNS untuk memetakan nama domain kustom pribadi ke nama host pribadinya. Anda memberikan ID DNS VPC Anda, dan ID zona yang dihosting yang Anda buat pada langkah sebelumnya. `DNSName`

```
{
  "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
        }
      }
    }
  ]
}
```

[change-resource-record-sets](#) Perintah berikut membuat catatan DNS untuk nama domain kustom pribadi:

```
aws route53 change-resource-record-sets \  
  --hosted-zone-id ZABCDEFG1234 \  
  --change-batch file://path/to/your/setup-dns-record.json
```

Ganti `hosted-zone-id` dengan ID Zona Dihosting Route 53 dari catatan DNS yang ditetapkan di akun Anda. Nilai `change-batch` parameter menunjuk ke file JSON.

Langkah selanjutnya untuk konsumen API

Anda sekarang dapat menjalankan API pribadi Anda sendiri Akun AWS. Di VPC Anda, Anda dapat menggunakan perintah `curl` berikut untuk mengakses nama domain kustom pribadi Anda.

```
curl https://private.example.com/v1
```

Untuk informasi selengkapnya tentang cara lain untuk menjalankan API pribadi Anda, lihat [the section called “Memanggil API pribadi menggunakan nama domain khusus”](#).

Konsumen API: Hapus asosiasi akses nama domain Anda dengan nama domain kustom pribadi

Jika Anda adalah konsumen API, kapan saja, Anda dapat menghapus sumber daya asosiasi akses. Penyedia API tidak dapat menghapus asosiasi akses nama domain untuk Anda.

Kami menyarankan agar Anda selalu menghapus asosiasi akses nama domain saat Anda tidak lagi menggunakannya.

Konsol Manajemen AWS

Untuk menghapus asosiasi akses nama domain

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Asosiasi akses nama domain.
3. Pilih asosiasi akses nama domain Anda, lalu pilih Hapus.
4. Konfirmasikan pilihan Anda, lalu pilih Hapus.

AWS CLI

`delete-access-association` Perintah berikut menghapus asosiasi akses:

```
aws apigateway delete-domain-name-access-association \  
  --domain-name domain-name \  
  --access-association-id access-association-id
```

```
--domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcsource/vpce-abcd1234efg'
```

Buat nama domain khusus untuk APIs penggunaan pribadi CloudFormation

Contoh CloudFormation template berikut membuat API pribadi dan nama domain kustom pribadi, memetakan API pribadi ke nama domain kustom, dan kemudian membuat asosiasi akses nama domain. Anda perlu menyediakan titik akhir VPC, nama domain, dan sertifikat ARN Anda sendiri.

Pertimbangan berikut dapat memengaruhi penggunaan Anda CloudFormation untuk membuat nama domain kustom pribadi:

- Anda tidak dapat menolak asosiasi akses nama domain menggunakan CloudFormation. Untuk menolak asosiasi akses nama domain, gunakan file. AWS CLI
- Gunakan `AWS::ApiGateway::DomainNameV2` CloudFormation properti untuk membuat nama domain kustom pribadi.
- Gunakan `AWS::ApiGateway::BasePathMappingV2` CloudFormation properti untuk membuat pemetaan jalur dasar.

```
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

```

Memanggil API pribadi

Anda hanya dapat menjalankan API pribadi dari dalam VPC menggunakan titik akhir VPC. API pribadi Anda harus memiliki kebijakan sumber daya yang memungkinkan titik akhir spesifik VPCs dan VPC menjalankan API Anda.

Jika Anda menjalankan API pribadi tanpa menggunakan nama domain khusus atau nama DNS pribadi dan nama domain APIs atau Anda menggunakan kebijakan keamanan yang dimulai `SecurityPolicy_`, Anda harus menyetel mode akses titik akhir. BASIC Untuk informasi selengkapnya, lihat [the section called “Mode akses titik akhir”](#).

Memanggil API pribadi menggunakan nama domain khusus

Untuk menjalankan API pribadi menggunakan nama domain khusus, titik akhir VPC Anda memerlukan asosiasi akses nama domain dengan nama domain khusus, dan nama domain khusus perlu mengizinkan akses ke titik akhir VPC untuk memanggilnya. Untuk informasi selengkapnya, lihat [the section called “Nama domain khusus untuk pribadi APIs”](#).

Tidak ada perbedaan antara memanggil nama domain kustom pribadi di VPC Anda Akun AWS sendiri atau yang berbeda. Akun AWS

Gunakan nama domain kustom Anda

Di dalam VPC Anda, Anda dapat memanggil API Anda menggunakan nama domain kustom. Contoh berikut adalah perintah curl untuk memanggil nama domain kustom pribadi Anda:

```
curl https://private.example.com
```

Gunakan nama host DNS pribadi khusus titik akhir

Anda dapat menjalankan API menggunakan nama domain khusus dan nama host DNS pribadi khusus titik akhir.

```
curl https://private-dns-hostname.execute-api.region.vpce.amazonaws.com/basepath -H  
'Host:custom-domain-name'
```

Contoh berikut adalah perintah curl untuk memanggil nama domain kustom Anda menggunakan nama host DNS pribadi khusus titik akhir:

```
curl https://vpce-123456-abc000.execute-api.us-east-2.vpce.amazonaws.com/test -H  
'Host:private.example.com'
```

Memanggil API pribadi tanpa menggunakan nama domain khusus

Untuk menjalankan API pribadi Anda tanpa menggunakan nama domain khusus, Anda perlu mengidentifikasi nama DNS untuk API Anda. Prosedur berikut menunjukkan cara menemukan nama DNS Anda.

Konsol Manajemen AWS

Untuk menemukan nama DNS

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di <https://console.aws.amazon.com/vpc/>
2. Di panel navigasi utama, pilih Endpoints lalu pilih endpoint VPC antarmuka Anda untuk API Gateway.
3. Di panel Detail, Anda akan melihat lima nilai di bidang nama DNS. Tiga yang pertama adalah nama DNS publik untuk API Anda. Dua lainnya adalah nama DNS pribadi untuk itu.

AWS CLI

Gunakan [describe-vpc-endpoints](#) perintah berikut untuk mencantumkan nilai DNS Anda.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

Tiga yang pertama adalah nama DNS publik untuk API Anda. Dua lainnya adalah nama DNS pribadi untuk itu.

Memanggil API pribadi menggunakan alias Route53

Anda dapat mengaitkan atau memisahkan titik akhir VPC dengan API pribadi Anda. Untuk informasi selengkapnya, lihat [the section called “\(Optional\) Kaitkan atau lepaskan titik akhir VPC dengan API pribadi”](#).

Setelah mengaitkan titik akhir VPC dengan API pribadi, Anda dapat menggunakan URL dasar berikut untuk menjalankan API:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Misalnya, jika Anda menyiapkan GET /pets metode untuk test stage, dan REST API ID Anda01234567ab, dan ID titik akhir VPC Andavpce-01234567abcdef012, dan Wilayah Anda, Anda dapat memanggil API Anda sebagaius-west-2:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/test/pets
```

Memanggil API pribadi menggunakan nama DNS pribadi

Jika Anda telah mengaktifkan DNS pribadi, Anda dapat mengakses API pribadi menggunakan nama DNS pribadi berikut:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

URL dasar untuk menjalankan API adalah dalam format berikut:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Misalnya, jika Anda menyiapkan GET /pets metode untuk test stage, dan REST API ID adalah 01234567ab dan Region Andaus-west-2, Anda dapat memanggil API pribadi Anda dengan memasukkan URL berikut di browser:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Atau, Anda dapat menggunakan perintah cURL berikut untuk menjalankan API pribadi Anda:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Warning

Jika Anda mengaktifkan DNS pribadi untuk titik akhir VPC Anda, Anda tidak akan dapat mengakses titik akhir default untuk publik. APIs Untuk informasi selengkapnya, lihat [Mengapa saya tidak dapat terhubung ke API publik saya dari titik akhir VPC API Gateway?](#) .

Memanggil API pribadi menggunakan Direct Connect

Anda dapat menggunakan Direct Connect untuk membuat koneksi pribadi khusus dari jaringan lokal ke Amazon VPC dan mengakses titik akhir API pribadi Anda melalui koneksi tersebut dengan menggunakan nama DNS publik.

Anda juga dapat menggunakan nama DNS pribadi untuk mengakses API pribadi Anda dari jaringan lokal dengan menyiapkan titik akhir Amazon Route 53 Resolver masuk dan meneruskan semua kueri DNS DNS pribadi dari jaringan jarak jauh Anda. Untuk informasi selengkapnya, lihat [Meneruskan kueri DNS masuk](#) ke Panduan VPCs Pengembang Amazon Route 53.

Memanggil API pribadi menggunakan nama host DNS publik khusus titik akhir

Anda dapat mengakses API pribadi menggunakan nama host DNS khusus titik akhir. Ini adalah nama host DNS publik yang berisi ID titik akhir VPC atau ID API untuk API pribadi Anda.

URL dasar yang dihasilkan dalam format berikut:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Misalnya, jika Anda menyiapkan GET /pets metode untuk test stage, dan REST API ID Anda abc1234, nama host DNS publiknya adalah vpce-def-01234567, dan Region Anda us-west-2, Anda dapat memanggil API pribadi menggunakan VPCe IDnya dengan menggunakan Host header dalam perintah cURL:

```
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'
```

Atau, Anda dapat memanggil API pribadi Anda melalui ID API-nya dengan menggunakan x-apigw-api-id header dalam perintah cURL dalam format berikut:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -H 'x-apigw-api-id:{api-id}'
```

Pantau REST APIs di API Gateway

Di bagian ini, Anda dapat mempelajari cara memantau API menggunakan CloudWatch metrik, CloudWatch Log, Firehose, dan AWS X-Ray Dengan menggabungkan log CloudWatch eksekusi dan CloudWatch metrik, Anda dapat mencatat kesalahan dan jejak eksekusi, serta memantau kinerja API

Anda. Anda mungkin juga ingin mencatat panggilan API ke Firehose. Anda juga dapat menggunakan AWS X-Ray untuk melacak panggilan melalui layanan hilir yang membentuk API Anda.

Note

API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:

- 413 Kesalahan Permintaan Entitas Terlalu Besar
- 431 Permintaan Bidang Header Kesalahan Terlalu Besar
- Berlebihan 429 Terlalu Banyak Kesalahan Permintaan
- 400 error seri dari permintaan yang dikirim ke domain kustom yang tidak memiliki pemetaan API
- 500 kesalahan seri yang disebabkan oleh kegagalan internal

API Gateway tidak akan menghasilkan log dan metrik saat menguji metode REST API. CloudWatch Entri disimulasikan. Lihat informasi yang lebih lengkap di [the section called “Gunakan konsol untuk menguji metode REST API”](#).

Topik

- [Pantau eksekusi REST API dengan CloudWatch metrik Amazon](#)
- [Siapkan CloudWatch logging untuk REST APIs di API Gateway](#)
- [Log panggilan REST API ke Amazon Data Firehose di API Gateway](#)
- [Variabel untuk pencatatan akses untuk API Gateway](#)
- [Lacak permintaan pengguna ke REST APIs menggunakan X-Ray di API Gateway](#)

Pantau eksekusi REST API dengan CloudWatch metrik Amazon

Anda dapat memantau eksekusi API dengan menggunakan CloudWatch, yang mengumpulkan dan memproses data mentah dari API Gateway menjadi metrik yang dapat dibaca. near-real-time Statistik ini dicatat untuk jangka waktu 15 bulan sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, data metrik API Gateway dikirim secara otomatis CloudWatch dalam periode satu menit. Untuk informasi selengkapnya, lihat [Apa itu Amazon CloudWatch?](#) di Panduan CloudWatch Pengguna Amazon.

Metrik yang dilaporkan oleh API Gateway memberikan informasi yang dapat Anda analisis dengan berbagai cara. Daftar berikut menunjukkan beberapa kegunaan umum untuk metrik yang merupakan saran untuk Anda mulai:

- Pantau `IntegrationLatency` metrik untuk mengukur respons backend.
- Pantau metrik Latensi untuk mengukur respons panggilan API Anda secara keseluruhan.
- Pantau `CacheHitCount` dan `CacheMissCount` metrik untuk mengoptimalkan kapasitas cache untuk mencapai kinerja yang diinginkan.

Topik

- [Dimensi dan metrik Amazon API Gateway](#)
- [Lihat CloudWatch metrik dengan dasbor API di API Gateway](#)
- [Lihat metrik API Gateway di konsol CloudWatch](#)
- [Melihat peristiwa log API Gateway di CloudWatch konsol](#)
- [Alat pemantauan AWS untuk API Gateway](#)

Dimensi dan metrik Amazon API Gateway

Metrik dan dimensi yang dikirimkan API Gateway ke Amazon CloudWatch tercantum di bawah ini. Untuk informasi selengkapnya, lihat [Pantau eksekusi REST API dengan CloudWatch metrik Amazon](#).

Metrik API Gateway

Amazon API Gateway mengirimkan data metrik ke CloudWatch setiap menit.

Namespace `AWS/ApiGateway` mencakup metrik berikut.

Metrik	Deskripsi
4XXError	Jumlah kesalahan sisi klien yang ditangkap dalam periode tertentu. API Gateway menghitung kode status respons gateway yang dimodifikasi sebagai 4 XXError kesalahan.

Metrik	Deskripsi
	SumStatistik mewakili metrik ini, yaitu, jumlah total 4XXError kesalahan dalam periode tertentu. AverageStatistik mewakili tingkat 4XXError kesalahan, yaitu, jumlah total 4XXError kesalahan dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
5XXError	Jumlah kesalahan sisi server yang ditangkap dalam periode tertentu. SumStatistik mewakili metrik ini, yaitu, jumlah total 5XXError kesalahan dalam periode tertentu. AverageStatistik mewakili tingkat 5XXError kesalahan, yaitu, jumlah total 5XXError kesalahan dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
CacheHitCount	Jumlah permintaan yang disajikan dari cache API dalam periode tertentu. SumStatistik mewakili metrik ini, yaitu, jumlah total hit cache pada periode tertentu. AverageStatistik mewakili tingkat hit cache, yaitu, jumlah total klik cache dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count

Metrik	Deskripsi
CacheMissCount	Jumlah permintaan yang disajikan dari backend dalam periode tertentu, saat caching API diaktifkan. SumStatistik mewakili metrik ini, yaitu, jumlah total cache yang hilang dalam periode tertentu. AverageStatistik mewakili tingkat kehilangan cache, yaitu, jumlah total cache yang hilang dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
Count	Jumlah total permintaan API dalam periode tertentu. SampleCount Statistik mewakili metrik ini. Unit: Count
IntegrationLatency	Waktu antara saat API Gateway menyampaikan permintaan ke backend dan saat menerima respons dari backend. Unit: Millisecond
Latency	Waktu antara saat API Gateway menerima permintaan dari klien dan saat mengembalikan respons ke klien. Latensi mencakup latensi integrasi dan overhead API Gateway lainnya. Unit: Millisecond

Dimensi untuk metrik

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

Note

API Gateway menghapus karakter non-ASCII dari ApiName dimensi sebelum mengirim metrik ke CloudWatch. Jika tidak, APIName mengandung karakter ASCII, API ID digunakan sebagai ApiName.

Dimensi	Deskripsi
ApiName	Memfilter metrik API Gateway untuk REST API dengan nama API yang ditentukan.
ApiName, Method, Resource, Stage	Memfilter metrik API Gateway untuk metode API dengan nama API, tahapan, sumber daya, dan metode yang ditentukan. API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Di konsol, pilih panggung, lalu untuk Log dan penelusuran, pilih Edit. Pilih Metrik terperinci, lalu pilih Simpan perubahan. Atau, Anda dapat memanggil AWS CLI perintah update-stage untuk memperbarui properti ke. <code>metricsEnabled true</code> Mengaktifkan metrik ini akan dikenakan biaya tambahan ke akun Anda. Untuk informasi harga, lihat CloudWatch Harga Amazon.
ApiName, Stage	Memfilter metrik API Gateway untuk sumber daya tahap API dengan nama dan tahapan API yang ditentukan.

Lihat CloudWatch metrik dengan dasbor API di API Gateway

Anda dapat menggunakan dasbor API di API Gateway Console untuk menampilkan CloudWatch metrik API yang Anda gunakan di API Gateway. Ini ditampilkan sebagai ringkasan aktivitas API dari waktu ke waktu.

Topik

- [Prasyarat](#)
- [Periksa aktivitas API di dasbor](#)

Prasyarat

1. Anda harus memiliki API yang dibuat di API Gateway. Ikuti petunjuk dalam [Kembangkan REST API di API Gateway](#).
2. Anda harus memiliki API yang diterapkan setidaknya sekali. Ikuti petunjuk dalam [Menerapkan REST APIs di API Gateway](#).

Periksa aktivitas API di dasbor

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API.
3. Di panel navigasi utama, pilih Dasbor.
4. Untuk Panggung, pilih tahap yang diinginkan.
5. Pilih Rentang tanggal untuk menentukan rentang tanggal.
6. Segarkan, jika diperlukan, dan lihat metrik individual yang ditampilkan dalam grafik terpisah berjudul Panggilan API, Latensi, Latensi integrasi, Latensi, kesalahan 4xx, dan kesalahan 5xx.

Tip

Untuk memeriksa CloudWatch metrik tingkat metode, pastikan Anda telah mengaktifkan CloudWatch Log pada tingkat metode. Untuk informasi selengkapnya tentang cara mengatur logging tingkat metode, lihat. [Ganti pengaturan tingkat panggung](#)

Lihat metrik API Gateway di konsol CloudWatch

Metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, lalu berdasarkan berbagai kombinasi dimensi dalam setiap namespace. Untuk melihat metrik di tingkat metode untuk API Anda, aktifkan metrik terperinci. Untuk informasi selengkapnya, lihat [the section called “Perbarui pengaturan panggung”](#).

Untuk melihat metrik API Gateway menggunakan konsol CloudWatch

1. Buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Jika perlu, ubah Wilayah AWS. Dari bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada.
3. Pada panel navigasi, silakan pilih Metrik.
4. Di tab Semua metrik, pilih API Gateway.
5. Untuk melihat metrik berdasarkan tahap, pilih panel Berdasarkan Tahap. Kemudian, pilih nama Anda APIs dan metrik.
6. Untuk melihat metrik berdasarkan API tertentu, pilih panel By Api Name. Kemudian, pilih nama Anda APIs dan metrik.

Untuk melihat metrik menggunakan CLI AWS

1. Gunakan perintah [daftar-metrik berikut untuk membuat daftar metrik](#):

```
aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
```

Setelah Anda membuat metrik, biarkan hingga 15 menit agar metrik muncul. Untuk melihat statistik metrik lebih cepat, gunakan [get-metric-data](#) atau [get-metric-statistics](#).

2. Gunakan [get-metrics-statistics](#) perintah berikut untuk melihat rata-rata selama periode waktu menggunakan interval 5 menit:

```
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
```

Melihat peristiwa log API Gateway di CloudWatch konsol

Bagian berikut menjelaskan prasyarat yang diperlukan dan cara melihat peristiwa log API Gateway di konsol. CloudWatch

Prasyarat

1. Anda harus memiliki API yang dibuat di API Gateway. Ikuti petunjuk dalam [Kembangkan REST API di API Gateway](#).
2. Anda harus memiliki API yang diterapkan dan dipanggil setidaknya sekali. Ikuti instruksi di [Menerapkan REST APIs di API Gateway](#) dan [Panggil REST APIs di API Gateway](#).
3. Anda harus mengaktifkan CloudWatch Log untuk sebuah panggung. Ikuti petunjuk dalam [Siapkan CloudWatch logging untuk REST APIs di API Gateway](#).

Untuk melihat permintaan dan tanggapan API yang dicatat menggunakan CloudWatch konsol

1. Buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Jika perlu, ubah Wilayah AWS. Dari bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada. Untuk informasi selengkapnya, lihat [Wilayah dan Titik Akhir](#).
3. Di panel navigasi, pilih Log, Grup log.
4. Di bawah tabel Grup Log, pilih grup log dari nama API-gateway-Execution-logs_ `{{stage-name}}`. rest-api-id
5. Di bawah tabel Aliran Log, pilih aliran log. Anda dapat menggunakan stempel waktu untuk membantu menemukan aliran log yang Anda minati.
6. Pilih Teks untuk melihat teks mentah atau pilih Baris untuk melihat acara baris demi baris.

Important

CloudWatch memungkinkan Anda menghapus grup log atau aliran. Jangan menghapus grup atau aliran log API Gateway API secara manual; biarkan API Gateway mengelola sumber daya ini. Menghapus grup log atau aliran secara manual dapat menyebabkan permintaan dan tanggapan API tidak dicatat. Jika itu terjadi, Anda dapat menghapus seluruh grup log untuk API dan menerapkan ulang API. Ini karena API Gateway membuat grup log atau aliran log untuk tahap API pada saat digunakan.

Alat pemantauan AWS untuk API Gateway

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau API Gateway. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda secara otomatis, sementara alat lain memerlukan intervensi manual. Sebaiknya Anda mengotomatisasi tugas pemantauan sebanyak mungkin.

Alat pemantauan otomatis di AWS

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton API Gateway dan melaporkan bila ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat [Pantau eksekusi REST API dengan CloudWatch metrik Amazon](#).
- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat [Apa itu CloudWatch Log?](#) di Panduan CloudWatch Pengguna Amazon.
- Amazon EventBridge (sebelumnya disebut CloudWatch Acara) — Cocokkan peristiwa dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge?](#) dalam EventBridge User Guide.
- AWS CloudTrail Pemantauan Log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat [Bekerja dengan File CloudTrail Log](#) di Panduan AWS CloudTrail Pengguna.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan API Gateway melibatkan pemantauan secara manual item yang tidak CloudWatch tercakup oleh alarm. API Gateway CloudWatch,, dan dasbor AWS konsol lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda juga memeriksa file log pada eksekusi API.

- Dasbor API Gateway menampilkan statistik berikut untuk tahap API tertentu selama periode waktu tertentu:
 - Panggilan API
 - Cache Hit, hanya ketika caching API diaktifkan.
 - Cache Miss, hanya ketika caching API diaktifkan.
 - Latensi
 - Latensi Integrasi
 - Kesalahan 4XX
 - Kesalahan 5XX
- CloudWatch Halaman beranda menunjukkan:
 - Alarm dan status saat ini
 - Grafik alarm dan sumber daya
 - Status kesehatan layanan

Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:

- Membuat [dasbor yang disesuaikan](#) untuk memantau layanan yang penting bagi Anda
- Data metrik grafik untuk memecahkan masalah dan mengungkap tren
- Cari dan telusuri semua metrik AWS sumber daya Anda
- Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah

Membuat CloudWatch alarm untuk memantau API Gateway

Anda dapat membuat CloudWatch alarm yang mengirimkan pesan Amazon SNS saat alarm berubah status. Alarm mengawasi satu metrik selama jangka waktu yang Anda tentukan, dan melakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama jangka waktu tertentu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Penskalaan Otomatis. Alarm memanggil tindakan untuk perubahan status berkelanjutan saja. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu.

Siapkan CloudWatch logging untuk REST APIs di API Gateway

Untuk membantu men-debug masalah yang terkait dengan eksekusi permintaan atau akses klien ke API Anda, Anda dapat mengaktifkan Amazon CloudWatch Logs untuk mencatat panggilan API. Untuk informasi lebih lanjut tentang CloudWatch, lihat [the section called “CloudWatch metrik”](#).

CloudWatch format log untuk API Gateway

Ada dua jenis log masuk API CloudWatch: logging eksekusi dan logging akses. Dalam pencatatan eksekusi, API Gateway mengelola CloudWatch Log. Prosesnya mencakup pembuatan grup log dan aliran log, dan pelaporan ke aliran log permintaan dan tanggapan pemanggil apa pun.

Data yang dicatat mencakup kesalahan atau jejak eksekusi (seperti nilai parameter permintaan atau respons atau muatan), data yang digunakan oleh otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus), apakah kunci API diperlukan, apakah paket penggunaan diaktifkan, dan informasi lainnya. API Gateway menyunting header otorisasi, nilai kunci API, dan parameter permintaan sensitif serupa dari data yang dicatat.

Untuk meningkatkan postur keamanan Anda, kami sarankan Anda menggunakan pencatatan eksekusi di INFO tingkat ERROR atau. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Saat Anda menerapkan API, API Gateway membuat grup log dan aliran log di bawah grup log. Grup log diberi nama mengikuti API-Gateway-Execution-Logs_{rest-api-id}/{stage_name} format. Dalam setiap grup log, log dibagi lagi menjadi aliran log, yang diurutkan berdasarkan Waktu Peristiwa Terakhir saat data yang dicatat dilaporkan.

Dalam pencatatan akses, Anda, sebagai pengembang API, ingin mencatat siapa yang telah mengakses API Anda dan bagaimana pemanggil mengakses API. Anda dapat membuat grup log Anda sendiri atau memilih grup log yang sudah ada yang dapat dikelola oleh API Gateway. Untuk menentukan detail akses, Anda memilih [\\$context](#) variabel, format log, dan tujuan grup log.

Format log akses harus menyertakan setidaknya `$context.requestId` atau `$context.extendedRequestId`. Sebagai praktik terbaik, sertakan `$context.requestId` dan `$context.extendedRequestId` dalam format log Anda.

`$context.requestId`

Ini mencatat nilai di `x-amzn-RequestId` header. Klien dapat mengganti nilai di `x-amzn-RequestId` header dengan nilai dalam format pengenalan unik universal (UUID). API Gateway mengembalikan ID permintaan ini di header `x-amzn-RequestId` respons. API Gateway menggantikan permintaan IDs yang diganti yang tidak dalam format UUID dengan log akses Anda. **UUID_REPLACED_INVALID_REQUEST_ID**

\$context.extendedRequestId

ExtendedRequestId adalah ID unik yang dihasilkan API Gateway. API Gateway mengembalikan ID permintaan ini di header `x-amz-apigw-id` respons. Pemanggil API tidak dapat memberikan atau mengganti ID permintaan ini. Anda mungkin perlu memberikan nilai ini ke AWS Support untuk membantu memecahkan masalah API Anda. Untuk informasi selengkapnya, lihat [the section called “Variabel untuk pencatatan akses untuk API Gateway”](#).

Pilih format log yang juga diadopsi oleh backend analitik Anda, seperti [Common Log Format](#) (CLF), JSON, XHTML, atau CSV. Anda kemudian dapat memasukkan log akses ke sana secara langsung agar metrik Anda dihitung dan dirender. [Untuk menentukan format log, atur grup log ARN pada properti accessLogSettings/destinationArn di atas panggung](#). Anda dapat memperoleh grup log ARN di konsol CloudWatch Untuk menentukan format log akses, atur format yang dipilih pada properti [accessLogSetting/format](#) di [panggung](#).

Contoh beberapa format log akses yang umum digunakan ditampilkan di konsol API Gateway dan dicantumkan sebagai berikut.

- CLF([Format Log Umum](#)):

```
$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(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,  
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,  
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Izin untuk CloudWatch pencatatan

Untuk mengaktifkan CloudWatch Log, Anda harus memberikan izin API Gateway untuk membaca dan menulis log CloudWatch untuk akun Anda. [Amazon APIGateway PushToCloudWatchLogs](#) memiliki semua izin yang diperlukan.

Note

API Gateway memanggil AWS Security Token Service untuk mengambil peran IAM, jadi pastikan itu AWS STS diaktifkan untuk Wilayah. Untuk informasi selengkapnya, lihat [Mengelola AWS STS di suatu AWS Wilayah](#).

Untuk memberikan izin ini ke akun Anda, buat peran IAM `apigateway.amazonaws.com` sebagai entitas tepercaya, lampirkan kebijakan sebelumnya ke peran IAM, dan tetapkan peran IAM ARN di properti `Arn` di Akun Anda. `cloudWatchRole` Anda harus mengatur properti `cloudWatchRoleArn` secara terpisah untuk setiap AWS Wilayah di mana Anda ingin mengaktifkan CloudWatch Log.

Jika Anda menerima kesalahan saat menyetel ARN peran IAM, periksa pengaturan akun AWS Security Token Service Anda untuk memastikan AWS STS bahwa diaktifkan di Wilayah yang Anda gunakan. Untuk informasi selengkapnya tentang mengaktifkan AWS STS, lihat [Mengelola AWS STS di AWS Wilayah](#) di Panduan Pengguna IAM.

Siapkan pencatatan CloudWatch API menggunakan konsol API Gateway

Untuk menyiapkan pencatatan CloudWatch API, Anda harus menerapkan API ke sebuah panggung. Anda juga harus mengonfigurasi ARN [peran CloudWatch Log yang sesuai](#) untuk akun Anda.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pada panel navigasi utama, pilih Pengaturan, lalu di bawah Logging, pilih Edit.

3. Untuk ARN peran CloudWatch log, masukkan ARN dari peran IAM dengan izin yang sesuai. Anda perlu melakukan ini sekali untuk setiap Akun AWS yang membuat APIs menggunakan API Gateway.
 4. Di panel navigasi utama, pilih APIs, lalu lakukan salah satu hal berikut:
 - a. Pilih API yang ada, lalu pilih panggung.
 - b. Buat API, lalu terapkan ke panggung.
 5. Di panel navigasi utama, pilih Tahapan.
 6. Di bagian Log dan penelusuran, pilih Edit.
 7. Untuk mengaktifkan pencatatan eksekusi:
 - a. Pilih level logging dari menu dropdown CloudWatch Log. Level logging adalah sebagai berikut:
 - Mati - Logging tidak diaktifkan untuk tahap ini.
 - Hanya kesalahan - Logging diaktifkan hanya untuk kesalahan.
 - Kesalahan dan info log - Logging diaktifkan untuk semua acara.
 - b. (Opsional) Pilih Penelusuran data untuk mengaktifkan pencatatan jejak data untuk tahap Anda. Ini dapat berguna untuk memecahkan masalah APIs, tetapi dapat mengakibatkan pencatatan data sensitif.
-  **Note**

Kami menyarankan Anda untuk tidak menggunakan penelusuran data untuk produksi APIs.
- c. (Opsional) Pilih Metrik terperinci untuk mengaktifkan CloudWatch metrik terperinci.

Untuk informasi selengkapnya tentang CloudWatch metrik, lihat [the section called “CloudWatch metrik”](#).

8. Untuk mengaktifkan pencatatan akses:
 - a. Aktifkan Pencatatan akses khusus.
 - b. Untuk akses log tujuan ARN, masukkan ARN dari grup log. Format ARN adalah.
`arn:aws:logs:{region}:{account-id}:log-group:log-group-name`

- c. Untuk Format Log, masukkan format log. Anda dapat memilih CLF, JSON, XHTML, atau CSV. Untuk mempelajari lebih lanjut tentang contoh format log, lihat [the section called "CloudWatch format log untuk API Gateway"](#).
9. Pilih Simpan perubahan.

Note

Anda dapat mengaktifkan pencatatan eksekusi dan mengakses logging secara independen satu sama lain.

API Gateway sekarang siap untuk mencatat permintaan ke API Anda. Anda tidak perlu menerapkan ulang API saat memperbarui pengaturan panggung, log, atau variabel tahap.

Siapkan pencatatan CloudWatch API menggunakan CloudFormation

Gunakan contoh CloudFormation template berikut untuk membuat grup CloudWatch log Amazon Logs dan mengonfigurasi eksekusi dan mengakses logging untuk sebuah panggung. Untuk mengaktifkan CloudWatch Log, Anda harus memberikan izin API Gateway untuk membaca dan menulis log CloudWatch untuk akun Anda. Untuk mempelajari selengkapnya, lihat [Mengaitkan akun dengan peran IAM](#) di Panduan AWS CloudFormation Pengguna.

```
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
```

Log panggilan REST API ke Amazon Data Firehose di API Gateway

Untuk membantu masalah debug yang terkait dengan akses klien ke API Anda, Anda dapat mencatat panggilan API ke Amazon Data Firehose. Untuk informasi selengkapnya tentang Firehose, lihat [Apa itu Amazon Data Firehose?](#) .

Untuk pencatatan akses, Anda hanya dapat mengaktifkan CloudWatch atau Firehose—Anda tidak dapat mengaktifkan keduanya. Namun, Anda dapat mengaktifkan CloudWatch pencatatan eksekusi dan Firehose untuk pencatatan akses.

Topik

- [Format log Firehose untuk API Gateway](#)
- [Izin untuk pencatatan Firehose](#)
- [Mengatur pencatatan akses Firehose dengan menggunakan konsol API Gateway](#)

Format log Firehose untuk API Gateway

[Firehose logging menggunakan format yang sama dengan logging. CloudWatch](#)

Izin untuk pencatatan Firehose

Saat pencatatan akses Firehose diaktifkan di panggung, API Gateway akan membuat peran terkait layanan di akun Anda jika peran tersebut belum ada. Peran tersebut dinamai `AWSServiceRoleForAPIGateway` dan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat [Menggunakan Peran Tertaut Layanan](#).

Note

Nama aliran Firehose Anda harus. `amazon-apigateway-{your-stream-name}`

Mengatur pencatatan akses Firehose dengan menggunakan konsol API Gateway

Untuk menyiapkan pencatatan API, Anda harus menerapkan API ke sebuah panggung. Anda juga harus membuat aliran Firehose.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Lakukan salah satu tindakan berikut:
 - a. Pilih API yang ada, lalu pilih panggung.
 - b. Buat API dan terapkan ke panggung.
3. Di panel navigasi utama, pilih Tahapan.
4. Di bagian Log dan penelusuran, pilih Edit.
5. Untuk mengaktifkan pencatatan akses ke aliran Firehose:
 - a. Aktifkan Pencatatan akses khusus.
 - b. Untuk akses log tujuan ARN, masukkan ARN dari aliran Firehose. Format ARN adalah.
`arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`
6. Pilih Simpan perubahan.

Note

Nama aliran Firehose Anda harus. `amazon-apigateway-{your-stream-name}`

- c. Untuk format Log, masukkan format log. Anda dapat memilih CLF, JSON, XHTML, atau CSV. Untuk mempelajari lebih lanjut tentang contoh format log, lihat [the section called "CloudWatch format log untuk API Gateway"](#).

API Gateway sekarang siap untuk mencatat permintaan ke API Anda ke Firehose. Anda tidak perlu menerapkan ulang API saat memperbarui pengaturan panggung, log, atau variabel tahap.

Variabel untuk pencatatan akses untuk API Gateway

Dalam pencatatan akses, Anda, sebagai pengembang API, ingin mencatat siapa yang telah mengakses API Anda dan bagaimana pemanggil mengakses API. Anda dapat membuat grup log Anda sendiri atau memilih grup log yang ada yang dapat dikelola oleh API Gateway. Untuk menentukan rincian akses, Anda dapat menggunakan variabel case-sensitive `$context` berikut.

Untuk daftar variabel referensi untuk transformasi data, lihat [the section called “Variabel untuk transformasi data”](#).

Parameter	Deskripsi
<code>\$context.accountId</code>	ID AWS akun pemilik API.
<code>\$context.apiId</code>	API Gateway identifier ditetapkan ke API Anda.
<code>\$context.authorize.error</code>	Pesan kesalahan otorisasi.
<code>\$context.authorize.latency</code>	Latensi otorisasi di ms.
<code>\$context.authorize.status</code>	Kode status dikembalikan dari upaya otorisasi.
<code>\$context.authorizer.claims. <i>property</i></code>	Properti klaim yang dikembalikan dari kumpulan pengguna Amazon Cognito setelah pemanggil metode berhasil diautentikasi. Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”.  Note Memanggil <code>\$context.authorizer.claims</code> mengembalikan null.
<code>\$context.authorizer.error</code>	Pesan kesalahan dikembalikan dari otorisasi.
<code>\$context.authorizer.integrationLatency</code>	Latensi integrasi otorisasi di ms.
<code>\$context.authorizer.integrationStatus</code>	Kode status dikembalikan dari otorisasi Lambda.
<code>\$context.authorizer.latency</code>	Latensi otorisasi di ms.
<code>\$context.authorizer.principalId</code>	Identifikasi pengguna utama yang terkait dengan token yang dikirim oleh klien dan

Parameter	Deskripsi
	dikembalikan dari otorisasi API Gateway Lambda (sebelumnya dikenal sebagai otorisasi khusus). Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda” .
<code>\$context.authorizer.<i>property</i></code>	Nilai stringifikasi dari pasangan nilai kunci <code>context</code> peta yang ditentukan dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan <code>context</code> peta berikut: <pre> "context" : { "key": "value", "numKey": 1, "boolKey": true } </pre> Memanggil <code>\$context.authorizer.key</code> mengembalikan "value" string, memanggil <code>\$context.authorizer.numKey</code> mengembalikan "1" string, dan memanggil <code>\$context.authorizer.boolKey</code> mengembalikan "true" string. Sebab <i>property</i>, satu-satunya karakter khusus yang didukung adalah (<code>_</code>) karakter garis bawah. Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda”.
<code>\$context.authorizer.requestId</code>	ID permintaan AWS titik akhir.
<code>\$context.authorizer.status</code>	Kode status dikembalikan dari otorisasi.
<code>\$context.authenticate.error</code>	Pesan kesalahan dikembalikan dari upaya otentikasi.

Parameter	Deskripsi
<code>\$context.authenticate.latency</code>	Latensi otentikasi di ms.
<code>\$context.authenticate.status</code>	Kode status dikembalikan dari upaya otentikasi.
<code>\$context.awsEndpointRequestId</code>	ID permintaan AWS titik akhir.
<code>\$context.cipherSuite</code>	Cipher, dalam format IANA, yang dinegosiasikan selama jabat tangan TLS antara klien dan API Gateway.
<code>\$context.customDomain.basePathMatched</code>	Jalur untuk pemetaan API yang cocok dengan permintaan masuk. Berlaku ketika klien menggunakan nama domain khusus untuk mengakses API. Misalnya jika klien mengirim permintaan ke <code>https://api.example.com/v1/orders/1234</code> , dan permintaan tersebut cocok dengan pemetaan API dengan jalur <code>v1/orders</code> , nilainya adalah <code>v1/orders</code> . Untuk mempelajari selengkapnya, lihat the section called “Pemetaan API” .
<code>\$context.customDomain.routingRuleIdMatched</code>	Aturan routing yang cocok dengan permintaan masuk. Berlaku ketika klien menggunakan nama domain khusus untuk mengakses API. Untuk mempelajari selengkapnya, lihat the section called “Aturan perutean” .
<code>\$context.deploymentId</code>	ID penerapan API.
<code>\$context.domainName</code>	Nama domain lengkap yang digunakan untuk memanggil API. Ini harus sama dengan Host header yang masuk.
<code>\$context.domainPrefix</code>	Label pertama dari <code>\$context.domainName</code> .
<code>\$context.endpointType</code>	Jenis titik akhir API.

Parameter	Deskripsi
<code>\$context.error.message</code>	String yang berisi pesan kesalahan API Gateway. Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponse pemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Untuk informasi selengkapnya, lihat the section called “Metrik-metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan” .
<code>\$context.error.messageString</code>	Nilai yang dikutip dari <code>\$context.error.message</code> , yaitu <code>"\$context.error.message"</code> .
<code>\$context.error.responseType</code>	Jenis GatewayResponse . Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponse pemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Untuk informasi selengkapnya, lihat the section called “Metrik-metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan” .
<code>\$context.error.validationErrorString</code>	Sebuah string yang berisi pesan kesalahan validasi rinci.
<code>\$context.extendedRequestId</code>	ID tambahan yang dibuat dan ditetapkan API Gateway ke permintaan API. ID permintaan yang diperluas berisi informasi yang berguna untuk debugging dan pemecahan masalah.

Parameter	Deskripsi
<code>\$context.httpMethod</code>	Metode HTTP yang digunakan . Nilai yang valid meliputi: DELETE, GET, HEAD, OPTIONS, PATCH, POST, dan PUT.
<code>\$context.identity.accountId</code>	ID AWS akun yang terkait dengan permintaan.
<code>\$context.identity.apiKey</code>	Untuk metode API yang memerlukan kunci API, variabel ini adalah kunci API yang terkait dengan permintaan metode. Untuk metode yang tidak memerlukan kunci API, variabel ini adalah null. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan” .
<code>\$context.identity.apiKeyId</code>	ID kunci API yang terkait dengan permintaan API yang memerlukan kunci API.
<code>\$context.identity.caller</code>	Pengenal utama penelepon yang menandatangani permintaan. Didukung untuk sumber daya yang menggunakan otorisasi IAM.

Parameter	Deskripsi
<code>\$context.identity.cognitoAuthenticationProvider</code>	Daftar dipisahkan koma dari semua penyedia otentikasi Amazon Cognito yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Untuk informasi tentang penyedia autentikasi Amazon Cognito yang tersedia, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Nilai yang mungkin termasuk <code>authenticated</code> untuk identitas yang diautentikasi dan <code>unauthenticated</code> untuk identitas yang tidak diautentikasi.
<code>\$context.identity.cognitoIdentityId</code>	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.

Parameter	Deskripsi
<code>\$context.identity.principalOrgId</code>	ID AWS organisasi .
<code>\$context.identity.sourceIp</code>	Alamat IP sumber koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Sertifikat PEM-encoded klien yang disajikan klien selama otentikasi TLS timbal balik. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.subjectDN</code>	Nama yang dibedakan dari subjek sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.issuerDN</code>	Nama terkemuka penerbit sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.serialNumber</code>	Nomor seri sertifikat. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Tanggal sebelum sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.

Parameter	Deskripsi
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Tanggal setelah sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.vpcId</code>	ID VPC VPC membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.vpcId</code>	ID titik akhir VPC dari titik akhir VPC membuat permintaan ke titik akhir API Gateway. Hadir hanya ketika Anda memiliki API pribadi.
<code>\$context.identity.user</code>	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk sumber daya yang menggunakan otorisasi IAM.
<code>\$context.identity.userAgent</code>	User-Agent Header pemanggil API.
<code>\$context.identity.userArn</code>	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.integration.error</code>	Pesan kesalahan dikembalikan dari integrasi.
<code>\$context.integration.integrationStatus</code>	Untuk integrasi proxy Lambda, kode status dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
<code>\$context.integration.latency</code>	Latensi integrasi dalam ms. Setara dengan <code>\$context.integrationLatency</code> .

Parameter	Deskripsi
<code>\$context.integration.requestId</code>	ID permintaan AWS titik akhir. Setara dengan <code>\$context.awsEndpointRequestId</code> .
<code>\$context.integration.responseTransferMode</code>	Mode transfer respons integrasi Anda. Ini bisa salah satu <code>BUFFERED</code> atau <code>STREAMED</code> .
<code>\$context.integration.status</code>	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda.
<code>\$context.integration.timeToAllHeaders</code>	Waktu antara saat API Gateway menetapkan koneksi integrasi saat menerima semua header respons integrasi dari klien.
<code>\$context.integration.timeToFirstContent</code>	Waktu antara saat API Gateway menetapkan koneksi integrasi saat menerima byte konten pertama.
<code>\$context.integrationLatency</code>	Latensi integrasi dalam ms.
<code>\$context.integrationStatus</code>	Untuk integrasi proxy Lambda, parameter ini mewakili kode status yang dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
<code>\$context.isCanaryRequest</code>	Kembali <code>true</code> jika permintaan diarahkan ke kenari dan <code>false</code> jika permintaan tidak diarahkan ke kenari. Hadir hanya ketika Anda mengaktifkan kenari.

Parameter	Deskripsi
<code>\$context.path</code>	Jalur permintaan. Misalnya, untuk URL permintaan non-proxy dari <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , <code>\$context.path</code> nilainya adalah <code>{stage}/root/child</code> .
<code>\$context.protocol</code>	Protokol permintaan, misalnya, HTTP/1.1.
	 Note API Gateway API dapat menerima HTTP/2 permintaan, tetapi API Gateway mengirimkan permintaan ke integrasi backend menggunakan HTTP/1.1 Akibatnya, protokol permintaan dicatat seolah-olah HTTP/1.1 klien mengirim permintaan yang menggunakan HTTP/2.
<code>\$context.requestId</code>	ID untuk permintaan. Klien dapat mengganti ID permintaan ini. Gunakan <code>\$context.extendedRequestId</code> untuk ID permintaan unik yang dihasilkan API Gateway.
<code>\$context.requestOverride.header. <i>header_name</i></code>	Header permintaan menimpa. Jika parameter ini didefinisikan, ini berisi header yang akan digunakan alih-alih Header HTTP yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .

Parameter	Deskripsi
<code>\$context.requestOverride.path.</code> <i>path_name</i>	Jalur permintaan menerima. Jika parameter ini ditentukan, parameter ini berisi jalur permintaan yang akan digunakan, bukan Parameter Jalur URL yang ditentukan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.requestOverride.querystring.</code> <i>querystring_name</i>	Permintaan query string override. Jika parameter ini didefinisikan, parameter ini berisi string permintaan yang akan digunakan, bukan Parameter String Kueri URL yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.responseLatency</code>	Latensi respons dalam ms.
<code>\$context.responseLength</code>	Panjang payload respon dalam byte.
<code>\$context.responseOverride.header.</code> <i>header_name</i>	Response header override. Jika parameter ini didefinisikan, ini berisi header yang akan dikembalikan, bukan header Response yang didefinisikan sebagai pemetaan Default di panel Integration Response. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .

Parameter	Deskripsi
<code>\$context.responseOverride.status</code>	Kode status respons menerima. Jika parameter ini didefinisikan, ini berisi kode status yang akan dikembalikan, bukan status respons Metode yang didefinisikan sebagai pemetaan Default di panel Respons Integrasi. Untuk informasi selengkapnya, lihat Ganti parameter permintaan dan respons API Anda serta kode status untuk REST API di API Gateway .
<code>\$context.requestTime</code>	Waktu permintaan yang diformat CLF (). dd/MMM/yyyy:HH:mm:ss +-hhmm
<code>\$context.requestTimeEpoch</code>	Waktu permintaan yang diformat Epoch , dalam milidetik.
<code>\$context.resourceId</code>	Pengidentifikasi yang ditetapkan API Gateway ke sumber daya Anda.
<code>\$context.resourcePath</code>	Jalan menuju sumber daya Anda. Misalnya, untuk URI permintaan non-proxy dari <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , <code>\$context.resourcePath</code> Nilainya adalah <code>/root/child</code> . Untuk informasi selengkapnya, lihat Tutorial: Membuat REST API dengan integrasi non-proxy HTTP .
<code>\$context.stage</code>	Tahap penerapan permintaan API (misalnya, Beta atau Prod).
<code>\$context.status</code>	Status respons metode.
<code>\$context.tlsVersion</code>	Versi TLS yang dinegosiasikan selama jabat tangan TLS antara klien dan API Gateway.

Parameter	Deskripsi
<code>\$context.waf.error</code>	Pesan kesalahan dikembalikan dari AWS WAF.
<code>\$context.waf.latency</code>	AWS WAF Latensi dalam ms.
<code>\$context.waf.status</code>	Kode status dikembalikan dari AWS WAF.
<code>\$context.xrayTraceId</code>	ID jejak untuk X-Ray jejak. Untuk informasi selengkapnya, lihat the section called “Mengatur AWS X-Ray” .
<code>\$context.wafResponseCode</code>	Tanggapan yang diterima dari AWS WAF : <code>WAF_ALLOW</code> atau <code>WAF_BLOCK</code> . Tidak akan diatur jika tahap tidak terkait dengan ACL web. Untuk informasi selengkapnya, lihat the section called “AWS WAF” .
<code>\$context.webaclArn</code>	ARN lengkap dari ACL web yang digunakan untuk memutuskan apakah akan mengizinkan atau memblokir permintaan. Tidak akan diatur jika tahap tidak terkait dengan ACL web. Lihat informasi yang lebih lengkap di the section called “AWS WAF” .

Lacak permintaan pengguna ke REST APIs menggunakan X-Ray di API Gateway

Anda dapat menggunakan [AWS X-Ray](#) untuk melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui Amazon API Gateway REST Anda APIs ke layanan yang mendasarinya. API Gateway mendukung penelusuran X-Ray untuk semua jenis titik akhir API API Gateway REST API: Regional, dioptimalkan tepi, dan pribadi. Anda dapat menggunakan X-Ray dengan Amazon API Gateway di semua AWS Wilayah di mana X-Ray tersedia.

Karena X-Ray memberi Anda end-to-end tampilan seluruh permintaan, Anda dapat menganalisis latensi di layanan backend Anda APIs dan mereka. Anda dapat menggunakan peta layanan X-Ray untuk melihat latensi seluruh permintaan dan layanan hilir yang terintegrasi dengan X-Ray. Anda juga

dapat mengonfigurasi aturan pengambilan sampel untuk memberi tahu X-Ray permintaan mana yang akan direkam dan berapa laju pengambilan sampel, sesuai dengan kriteria yang Anda tentukan.

Jika Anda memanggil API Gateway API dari layanan yang sudah dilacak, API Gateway meneruskan jejak, bahkan jika penelusuran X-Ray tidak diaktifkan di API.

Anda dapat mengaktifkan X-Ray untuk tahap API dengan menggunakan konsol API Gateway, atau dengan menggunakan API Gateway API atau CLI.

Topik

- [Siapkan AWS X-Ray dengan API Gateway REST APIs](#)
- [Menggunakan peta AWS X-Ray layanan dan melacak tampilan dengan API Gateway](#)
- [Konfigurasi aturan AWS X-Ray pengambilan sampel untuk API Gateway APIs](#)
- [AWS X-Ray jejak untuk Amazon API Gateway APIs](#)

Siapkan AWS X-Ray dengan API Gateway REST APIs

Di bagian ini Anda dapat menemukan informasi terperinci tentang cara mengatur [AWS X-Ray](#) dengan API Gateway REST APIs.

Topik

- [Mode penelusuran X-Ray untuk API Gateway](#)
- [Izin untuk penelusuran X-Ray](#)
- [Mengaktifkan penelusuran X-Ray di konsol API Gateway](#)
- [Mengaktifkan AWS X-Ray penelusuran menggunakan API Gateway CLI](#)

Mode penelusuran X-Ray untuk API Gateway

Jalur permintaan melalui aplikasi Anda dilacak dengan ID jejak. Jejak mengumpulkan semua segmen yang dihasilkan oleh satu permintaan, biasanya permintaan HTTP GET atau POST permintaan.

Ada dua mode penelusuran untuk API Gateway API:

- Pasif: Ini adalah pengaturan default jika Anda belum mengaktifkan penelusuran X-Ray pada tahap API. Pendekatan ini berarti API Gateway API hanya dilacak jika X-Ray telah diaktifkan pada layanan upstream.

- **Aktif:** Saat tahap API Gateway API memiliki pengaturan ini, API Gateway secara otomatis mengambil sampel permintaan pemanggilan API, berdasarkan algoritma pengambilan sampel yang ditentukan oleh X-Ray.

Saat penelusuran aktif diaktifkan di panggung, API Gateway akan membuat peran terkait layanan di akun Anda, jika peran tersebut belum ada. Peran tersebut diberi nama `AWSServiceRoleForAPIGateway` dan akan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat [Menggunakan Peran Tertaut Layanan](#).

Note

X-Ray menerapkan algoritma sampling untuk memastikan bahwa penelusuran efisien, sambil tetap memberikan sampel representatif dari permintaan yang diterima API Anda. Algoritma pengambilan sampel default adalah 1 permintaan per detik, dengan 5 persen permintaan sampel melewati batas itu.

Anda dapat mengubah mode penelusuran untuk API Anda dengan menggunakan konsol manajemen API Gateway, API Gateway CLI, atau AWS SDK.

Izin untuk penelusuran X-Ray

Saat Anda mengaktifkan penelusuran X-Ray di panggung, API Gateway akan membuat peran terkait layanan di akun Anda, jika peran tersebut belum ada. Peran tersebut diberi nama `AWSServiceRoleForAPIGateway` dan akan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat [Menggunakan Peran Tertaut Layanan](#).

Mengaktifkan penelusuran X-Ray di konsol API Gateway

Anda dapat menggunakan konsol Amazon API Gateway untuk mengaktifkan penelusuran aktif pada tahap API.

Langkah-langkah ini mengasumsikan bahwa Anda telah menerapkan API ke suatu panggung.

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API Anda, lalu di panel navigasi utama, pilih Tahapan.
3. Di panel Tahapan, pilih panggung.

4. Di bagian Log dan penelusuran, pilih Edit.
5. Untuk mengaktifkan penelusuran X-Ray aktif, pilih penelusuran X-Ray untuk mengaktifkan penelusuran X-Ray.
6. Pilih Simpan perubahan.

Setelah mengaktifkan X-Ray untuk tahap API, Anda dapat menggunakan konsol manajemen X-Ray untuk melihat jejak dan peta layanan.

Mengaktifkan AWS X-Ray penelusuran menggunakan API Gateway CLI

Perintah [create-stage](#) berikut membuat tahapan dengan penelusuran X-Ray aktif:

```
aws apigateway create-stage \  
  --rest-api-id rest-api-id \  
  --stage-name stage-name \  
  --deployment-id deployment-id \  
  --region region \  
  --tracing-enabled=true
```

Outputnya akan terlihat seperti berikut:

```
{  
  "tracingEnabled": true,  
  "stageName": stage-name,  
  "cacheClusterEnabled": false,  
  "cacheClusterStatus": "NOT_AVAILABLE",  
  "deploymentId": deployment-id,  
  "lastUpdatedDate": 1533849811,  
  "createdDate": 1533849811,  
  "methodSettings": {}  
}
```

Perintah [create-stage](#) berikut membuat panggung tanpa penelusuran X-Ray aktif:

```
aws apigateway create-stage \  
  --rest-api-id rest-api-id \  
  --stage-name stage-name \  
  --deployment-id deployment-id \  
  --region region \  
  --tracing-enabled=false
```

```
--tracing-enabled=false
```

Outputnya akan terlihat seperti berikut:

```
{
  "tracingEnabled": false,
  "stageName": stage-name,
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": deployment-id,
  "lastUpdatedDate": 1533849811,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```

[Tahap pembaruan berikut mengaktifkan penelusuran](#) X-Ray aktif untuk API yang diterapkan:

```
aws apigateway update-stage \
  --rest-api-id rest-api-id \
  --stage-name stage-name \
  --patch-operations op=replace,path=/tracingEnabled,value=true
```

[Tahap pembaruan berikut menonaktifkan penelusuran](#) X-Ray aktif untuk API yang diterapkan:

```
aws apigateway update-stage \
  --rest-api-id rest-api-id \
  --stage-name stage-name \
  --region region \
  --patch-operations op=replace,path=/tracingEnabled,value=false
```

Outputnya akan terlihat seperti berikut:

```
{
  "tracingEnabled": false,
  "stageName": stage-name,
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": deployment-id,
  "lastUpdatedDate": 1533850033,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```



```
}
```

Setelah Anda mengaktifkan X-Ray untuk tahap API Anda, gunakan X-Ray CLI untuk mengambil informasi jejak. Untuk informasi selengkapnya, lihat [Menggunakan X-Ray API dengan AWS CLI](#).

Menggunakan peta AWS X-Ray layanan dan melacak tampilan dengan API Gateway

Di bagian ini Anda dapat menemukan informasi terperinci tentang cara menggunakan peta [AWS X-Ray](#) layanan dan melacak tampilan dengan API Gateway.

Topik

- [Contoh peta layanan X-Ray](#)
- [Contoh tampilan jejak X-Ray](#)

Contoh peta layanan X-Ray

AWS X-Ray peta layanan menampilkan informasi tentang API Anda dan semua layanan hilirnya. Saat X-Ray diaktifkan untuk tahap API di API Gateway, Anda akan melihat node di peta layanan yang berisi informasi tentang keseluruhan waktu yang dihabiskan dalam layanan API Gateway. Anda bisa mendapatkan informasi terperinci tentang status respons dan histogram waktu respons API untuk jangka waktu yang dipilih. Untuk APIs mengintegrasikan dengan AWS layanan seperti AWS Lambda dan Amazon DynamoDB, Anda akan melihat lebih banyak node yang menyediakan metrik kinerja yang terkait dengan layanan tersebut. Akan ada peta layanan untuk setiap tahap API.

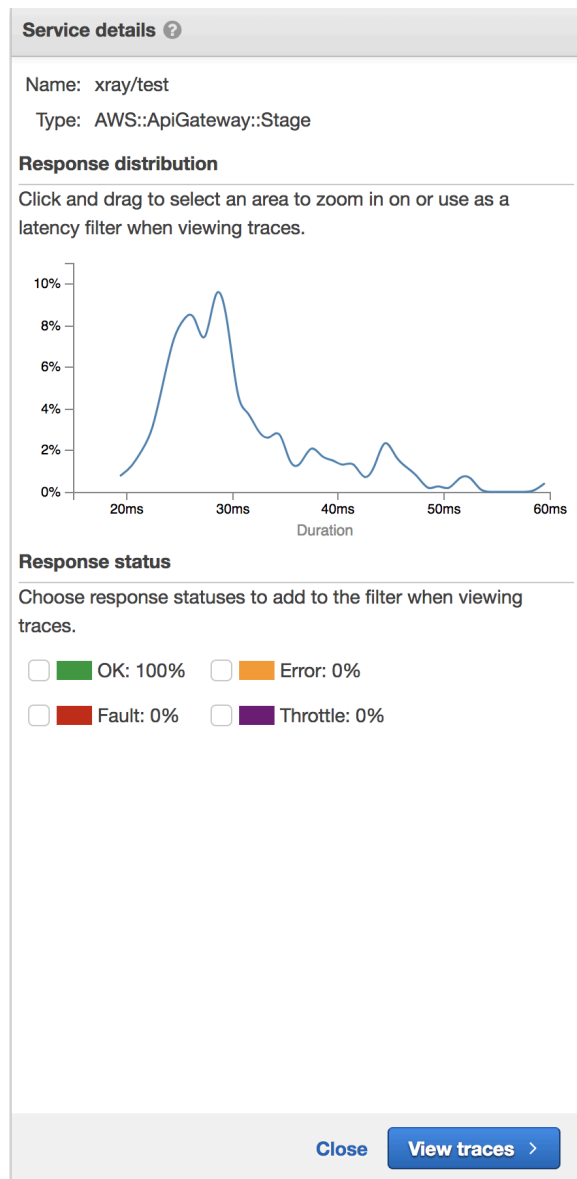
Contoh berikut menunjukkan peta layanan untuk test tahap API yang disebut `xray`. API ini memiliki dua integrasi Lambda. Node mewakili layanan API Gateway dan dua fungsi Lambda.

Untuk penjelasan rinci tentang struktur peta layanan, lihat [Menggunakan peta jejak X-Ray](#).



Dari peta layanan, Anda dapat memperbesar untuk melihat tampilan jejak tahap API Anda. Penelusuran akan menampilkan informasi mendalam mengenai API Anda, yang direpresentasikan sebagai segmen dan subsegmen. Misalnya, jejak untuk peta layanan yang ditunjukkan di atas akan mencakup segmen untuk layanan Lambda dan fungsi Lambda. Untuk informasi lebih lanjut, lihat [AWS Lambda dan AWS X-Ray](#).

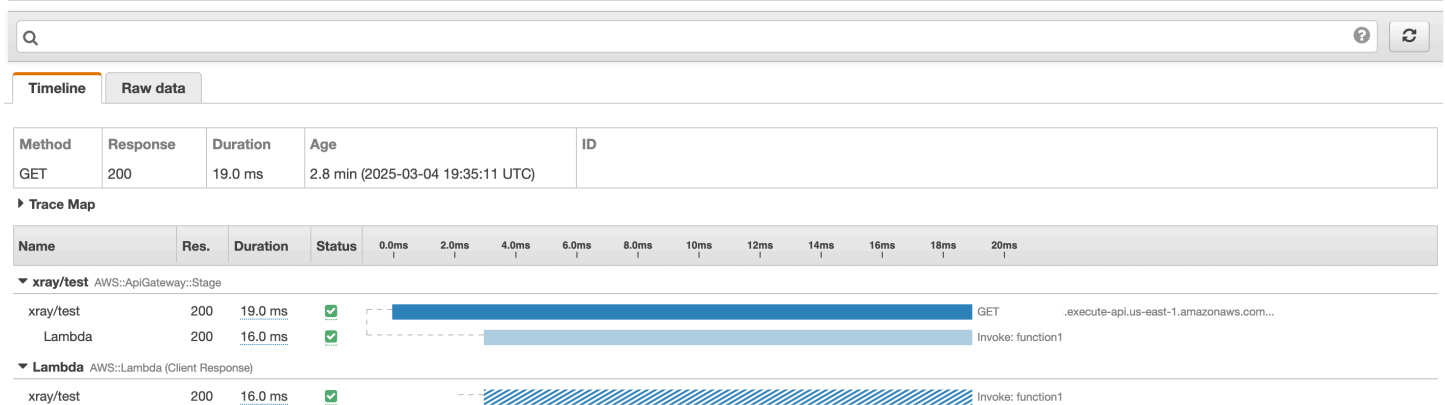
Jika Anda memilih node atau edge pada peta layanan X-Ray, konsol X-Ray menunjukkan histogram distribusi latensi. Anda dapat menggunakan histogram latensi untuk melihat berapa lama waktu yang dibutuhkan layanan untuk menyelesaikan permintaannya. Berikut ini adalah histogram tahap API Gateway yang disebutkan `xray/test` di peta layanan sebelumnya. Untuk penjelasan rinci tentang histogram distribusi latensi, lihat [Gunakan](#) Histogram Latensi.



Contoh tampilan jejak X-Ray

Diagram berikut menunjukkan tampilan jejak yang dihasilkan untuk contoh API yang dijelaskan di atas, dengan fungsi backend Lambda. Permintaan metode API yang berhasil ditampilkan dengan kode respons 200.

Untuk penjelasan rinci tentang tampilan jejak, lihat [Melihat jejak dan detail jejak](#).

[Traces](#) > Details

Konfigurasi aturan AWS X-Ray pengambilan sampel untuk API Gateway APIs

Anda dapat menggunakan AWS X-Ray konsol atau SDK untuk mengonfigurasi aturan pengambilan sampel untuk Amazon API Gateway API. Aturan sampling menentukan permintaan X-Ray mana yang harus direkam untuk API Anda. Dengan menyesuaikan aturan pengambilan sampel, Anda dapat mengontrol jumlah data yang Anda catat, dan mengubah perilaku pengambilan sampel dengan cepat tanpa mengubah atau men-deploy ulang kode Anda.

Sebelum Anda menentukan aturan pengambilan sampel X-Ray, baca topik berikut di Panduan Pengembang X-Ray:

- [Konfigurasi aturan pengambilan sampel](#)
- [Menggunakan Aturan Sampling dengan X-Ray API](#)

Topik

- [Nilai opsi aturan pengambilan sampel X-Ray untuk API Gateway APIs](#)
- [Contoh aturan pengambilan sampel X-Ray](#)

Nilai opsi aturan pengambilan sampel X-Ray untuk API Gateway APIs

Opsi pengambilan sampel X-Ray berikut relevan untuk API Gateway. Nilai string dapat menggunakan wildcard untuk mencocokkan satu karakter (?) atau nol atau lebih karakter (*). Untuk detail selengkapnya, termasuk penjelasan rinci tentang cara pengaturan Reservoir dan Rate digunakan, [Konfigurasi aturan pengambilan sampel](#).

- Rule name (string) — Sebuah nama unik untuk aturan.

- Prioritas (integer antara 1 dan 9999) — Prioritas aturan sampling. Layanan mengevaluasi aturan dalam urutan prioritas naik, dan membuat keputusan pengambilan sampel dengan aturan pertama yang cocok.
- Reservoir (bilangan bulat nonnegatif) — Sejumlah permintaan pencocokan yang tetap ke instrumen per detik, sebelum menerapkan tarif tetap. Reservoir tidak digunakan secara langsung oleh layanan, tetapi berlaku untuk semua layanan yang menggunakan aturan secara kolektif.
- Tarif (angka antara 0 dan 100) — Persentase permintaan yang cocok dengan instrumen, setelah reservoir habis.
- Nama layanan (string) - nama tahap API, dalam bentuk **{api-name}/{stage-name}**. Misalnya, jika Anda menerapkan API [PetStores](#) sampel ke tahap bernama `test`, nilai nama Layanan yang akan ditentukan dalam aturan pengambilan sampel Anda adalah **pets/test**
- Jenis layanan (string) - Untuk API Gateway API, baik **AWS::ApiGateway::Stage** atau **AWS::ApiGateway::*** dapat ditentukan.
- Host (string) — Nama host dari header host HTTP. Setel ini * agar cocok dengan semua nama host. Atau Anda dapat menentukan nama host penuh atau sebagian untuk dicocokkan, misalnya, **api.example.com** atau ***.example.com**.
- Resource ARN (string) — ARN dari tahap API, misalnya, **arn:aws:apigateway:region::restapis/api-id/stages/stage-name**

Nama panggung dapat diperoleh dari konsol atau API Gateway CLI atau API. Untuk informasi selengkapnya tentang format ARN, lihat. [Referensi Umum Amazon Web Services](#)

- Metode HTTP (string) — Metode yang akan diambil sampelnya, misalnya, **GET**.
- Jalur URL (string) — Jalur URL permintaan.
- (opsional) Atribut (kunci dan nilai) — Header dari permintaan HTTP asli, misalnya,, **ConnectionContent-Length**, atau **Content-Type**. Setiap nilai atribut dapat mencapai 32 karakter.

Contoh aturan pengambilan sampel X-Ray

Contoh aturan pengambilan sampel #1

Aturan ini mengambil sampel semua GET permintaan untuk `testxray` API pada test tahap tersebut.

- Nama aturan - **test-sampling**
- Prioritas - **17**

- Ukuran waduk — **10**
- Tarif tetap - **10**
- Nama layanan - **testxray/test**
- Jenis layanan - **AWS::ApiGateway::Stage**
- Metode HTTP - **GET**
- Sumber daya ARN — *
- Tuan rumah — *

Contoh aturan pengambilan sampel #2

Aturan ini mengambil sampel semua permintaan untuk testxray API pada prod tahap tersebut.

- Nama aturan - **prod-sampling**
- Prioritas - **478**
- Ukuran waduk — **1**
- Tarif tetap - **60**
- Nama layanan - **testxray/prod**
- Jenis layanan - **AWS::ApiGateway::Stage**
- Metode HTTP - *
- Sumber daya ARN — *
- Tuan rumah — *
- Atribut — **{}**

AWS X-Ray jejak untuk Amazon API Gateway APIs

Bagian ini membahas segmen AWS X-Ray jejak, subsegmen, dan bidang jejak lainnya untuk Amazon API Gateway. APIs

Sebelum Anda membaca bagian ini, tinjau topik-topik berikut di Panduan Pengembang X-Ray:

- [Gunakan sebuah Konsol Manajemen AWS](#)
- [Dokumen segmen X-Ray](#)
- [Konsep](#)

Topik

- [Contoh objek pelacakan untuk API Gateway API](#)
- [Memahami jejak](#)

Contoh objek pelacakan untuk API Gateway API

Bagian ini membahas beberapa objek yang mungkin Anda lihat dalam jejak untuk API Gateway API.

Anotasi

Anotasi dapat muncul di segmen dan subsegmen. Mereka digunakan sebagai ekspresi penyaringan dalam aturan pengambilan sampel untuk memfilter jejak. Untuk informasi selengkapnya, lihat [Mengonfigurasi aturan pengambilan sampel](#).

Berikut ini adalah contoh annotations objek, di mana tahap API diidentifikasi oleh ID API dan nama tahap API:

```
"annotations": {
  "aws:api_id": "a1b2c3d4e5",
  "aws:api_stage": "dev"
}
```

Untuk informasi selengkapnya tentang anotasi, lihat [dokumen segmen X-Ray](#), lalu pilih dokumen segmen X-Ray, Anotasi.

AWS data sumber daya

awsObjek hanya muncul di segmen. Berikut ini adalah contoh dari sebuah aws objek yang cocok dengan aturan Default sampling. Untuk penjelasan mendalam tentang aturan pengambilan sampel, lihat [Mengonfigurasi](#) aturan pengambilan sampel.

```
"aws": {
  "xray": {
    "sampling_rule_name": "Default"
  },
  "api_gateway": {
    "account_id": "123412341234",
    "rest_api_id": "a1b2c3d4e5",
    "stage": "dev",
    "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
```

```
}  
}
```

Untuk informasi selengkapnya tentang aws objek, lihat [dokumen segmen X-Ray](#), lalu pilih dokumen segmen X-Ray, data AWS sumber daya.

Memahami jejak

Berikut ini adalah segmen jejak untuk tahap API Gateway. Untuk penjelasan rinci tentang bidang yang membentuk segmen jejak, lihat [dokumen segmen X-Ray](#).

```
{  
  "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"
}

```

Portal API Gateway

Anda dapat menggunakan portal API Gateway untuk membagikan dokumentasi Anda APIs dan API dengan konsumen API di halaman web. Sebuah portal berisi satu set produk portal. Setiap produk portal adalah pengelompokan logis REST APIs dan berisi dokumentasi yang Anda buat dan publikasikan untuk konsumen API Anda. Halaman produk dalam portal berisi dokumentasi khusus di tingkat produk portal. Halaman endpoint REST produk berisi dokumentasi untuk masing-masing REST APIs dan jalur, metode, dan tahap REST API. Kombinasi halaman produk dan halaman akhir REST produk menyediakan dokumentasi lengkap untuk memungkinkan konsumen API mempelajari cara mulai menggunakan REST APIs Anda.

Gunakan portal untuk melakukan hal berikut:

- Gunakan API Gateway untuk seluruh siklus hidup API Anda, mulai dari membuat hingga mendokumentasikan dan APIs mendistribusikannya.
- Bagikan produk portal di seluruh AWS akun.
- Sesuaikan dan bagikan lokasi pusat untuk produk portal Anda dan sediakan halaman produk.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan portal API Gateway Anda:

- Anda hanya dapat membuat portal untuk berbagi REST APIs.
- Portal dibuat di tingkat AWS Wilayah. Portal Anda hanya dapat berisi REST APIs yang berada di Wilayah yang sama tempat Anda membuat portal.
- Anda hanya dapat menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS SDKs untuk membuat dan berbagi portal Anda.
- Anda hanya dapat mengontrol akses ke portal Anda menggunakan kumpulan pengguna Amazon Cognito. Konsumen portal Anda dapat masuk langsung melalui kumpulan pengguna, atau mereka dapat bergabung melalui penyedia identitas pihak ketiga (iDP). Jika Anda menggunakan kumpulan pengguna Amazon Cognito, Anda tidak dapat menyetel rahasia klien untuk kumpulan pengguna Anda. Untuk informasi selengkapnya, lihat [kumpulan pengguna Amazon Cognito](#).

Produk portal di API Gateway

Produk portal mewakili layanan atau fungsionalitas yang ingin Anda bagikan. Produk portal Anda adalah kumpulan titik akhir REST produk dan halaman produk. Titik akhir REST produk adalah titik akses ke produk portal Anda, dan mereka terdiri dari jalur dan metode REST API dan tahap penerapannya. Halaman produk adalah dokumentasi yang Anda berikan untuk menjelaskan bagaimana konsumen API dapat menggunakan titik akhir produk Anda. Produk portal dapat berisi seluruh Prod tahap sebagai kumpulan titik akhir REST produk atau hanya GET /pets sumber daya yang dikerahkan ke Prod tahap sebagai titik akhir REST produk tunggal.

Produk portal Anda dapat disesuaikan. Anda dapat menambahkan dokumentasi khusus, mengganti nama titik akhir REST produk, mengatur ulang urutan tampilan, menambahkan bagian baru, dan berbagi produk di seluruh akun. AWS Agar perubahan apa pun yang Anda buat pada produk portal Anda berlaku, Anda harus menerbitkan ulang portal apa pun yang menggunakan produk portal Anda.

Contoh produk portal adopsi hewan peliharaan

Sebagai contoh, Anda dapat memiliki beberapa REST APIs yang mewakili layanan adopsi hewan peliharaan. Anda dapat menggunakan API Gateway untuk membuat produk pet adoption portal. Produk portal ini akan membantu pelanggan menemukan mana yang harus APIs mereka gunakan untuk bertemu dan mengadopsi hewan peliharaan. Produk portal ini menggunakan REST APIs yang sudah Anda buat, tetapi memungkinkan Anda untuk berkumpul kembali dan mengaturnya. Anda juga dapat memberikan dokumentasi tentang syarat dan ketentuan penggunaan produk portal adopsi hewan peliharaan Anda dan membiarkan pelanggan mencobanya APIs. Informasi ini semua disimpan dalam produk portal Anda.

Tabel berikut menunjukkan tiga APIs yang mewakili produk portal adopsi hewan peliharaan, dan nama operasi titik akhir REST produk terkait dan nama bagian halaman.

SISA API ID	Jalur dan metode REST API	Tahap REST API	Nama operasi	Bagian halaman
kf5387miad	GET /dogs	Prod	View dogs	AdoptAnim als
kf5387miad	GET /dogs/ {dogId}	Prod	View dog	AdoptAnim als

SISA API ID	Jalur dan metode REST API	Tahap REST API	Nama operasi	Bagian halaman
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

Dalam contoh ini, dua REST APIs, ra8obxcevg dan kf5387miad, dikelompokkan bersama di AdoptAnimals bagian. Hasil navigasi pemesanan ini akan terlihat seperti berikut di portal:

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)

Karena ANY metode ini adalah metode catch-all, portal menampilkan semua metode HTTP yang didukung. Portal ini juga berisi dokumentasi yang dibuat oleh pemilik produk portal.

Langkah selanjutnya

Untuk memulai dengan produk portal, Anda dapat melakukan hal berikut:

- Untuk membuat produk portal, lihat [the section called “Buat produk portal”](#).
- Untuk mempelajari tentang fungsi try it, lihat [the section called “Aktifkan coba untuk titik akhir REST produk di portal Anda”](#).
- Untuk mempelajari tentang halaman produk, lihat [the section called “Buat halaman produk”](#).
- Untuk mempelajari tentang berbagi produk portal Anda, lihat [the section called “Bagikan produk portal”](#).

Setelah Anda membuat produk portal, Anda dapat mempublikasikannya ke portal. Lihat informasi yang lebih lengkap di [the section called “Buat portal”](#).

Membuat produk portal di API Gateway

Prosedur berikut menunjukkan cara membuat produk portal. Portal adalah kumpulan produk portal. Setelah Anda membuat produk portal Anda, Anda membuat titik akhir REST produk dan halaman produk. Untuk mempelajari tentang produk portal, lihat [the section called “Produk portal”](#).

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi cara Anda membuat produk portal:

- Produk portal Anda dapat berisi REST pribadi dan publik APIs. Private APIs tidak didukung untuk fungsi try it dan, sebagai hasilnya, memiliki perbedaan visual di portal Anda. Sebagai pemilik portal, Anda mungkin perlu memberikan dokumentasi untuk menjelaskan hal ini.
- Jika Anda membuat produk portal menggunakan AWS CLI atau AWS SDKs, portal Anda tidak akan memiliki titik akhir produk atau halaman produk. Anda perlu menambahkan sumber daya ini menggunakan konsol AWS CLI atau. Untuk mempelajari cara membuat endpoint REST produk, lihat [the section called “Buat titik akhir REST produk”](#). Untuk mempelajari cara membuat halaman produk, lihat [the section called “Buat halaman produk”](#).

Buat produk portal

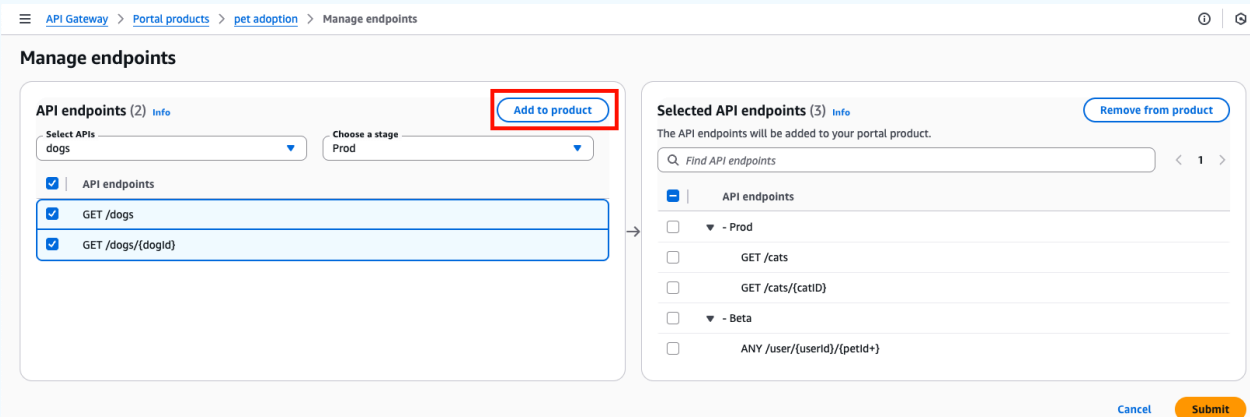
Prosedur berikut menunjukkan cara membuat produk portal.

Untuk membuat produk portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih Buat produk.
4. Untuk nama Produk, masukkan nama produk portal Anda.
5. Untuk deskripsi Produk, masukkan deskripsi.
6. Pilih Berikutnya.
7. Untuk memilih titik akhir REST produk Anda, di bawah titik akhir API pilih API, lalu pilih tahapan.
8. Untuk menambahkan titik akhir ke titik akhir REST produk Anda, pilih titik akhir API, lalu pilih Tambahkan ke produk.

Note

Jangan memilih Berikutnya tanpa terlebih dahulu memilih Tambahkan ke produk.



Titik akhir API akan muncul di daftar titik akhir API Terpilih.

9. Pilih Berikutnya.

10. Tinjau pilihan Anda dan pilih Buat produk.

Setelah Anda membuat produk portal menggunakan konsol, semua halaman produk dan halaman titik akhir REST produk Anda adalah draf dan tidak akan muncul di portal. Agar halaman produk dan halaman akhir REST produk Anda terlihat oleh konsumen, Anda perlu menambahkan draf Anda ke bagian. Jika Anda membuat produk portal Anda menggunakan AWS CLI atau AWS SDKs, Anda menambahkan draf ke bagian dalam AWS CLI perintah. Terlepas dari bagaimana Anda menambahkan draf Anda ke bagian, Anda harus mempublikasikan portal yang menggunakan produk portal Anda agar dapat dilihat oleh konsumen.

Untuk menambahkan draf Anda ke bagian halaman

1. Draf Anda tercantum di tab Dokumentasi. Ada halaman dokumentasi Draf untuk halaman produk Anda dan halaman referensi Draft API untuk halaman titik akhir REST produk Anda. Pilih halaman referensi Draft API.
2. Pilih halaman referensi API draf.

Jika Anda tidak memiliki titik akhir REST produk, Anda tidak akan memiliki halaman referensi API draf apa pun. Untuk mempelajari cara membuat endpoint REST produk, lihat [the section called “Buat titik akhir REST produk”](#).

3. Pilih Edit halaman.
4. Di halaman ini, Anda dapat menimpa bagian dokumentasi API yang ada atau menggunakan dokumentasi API Gateway. Agar konten halaman akhir REST produk Anda dapat dilihat oleh konsumen, di bawah nama Bagian, masukkan nama. Jika ini adalah /pets-GET titik akhir, nama halaman bisa Pets jadi.
5. Pilih Simpan perubahan.
6. Nama halaman baru yang Anda buat muncul di bagian halaman referensi API.

Agar halaman baru dapat dilihat oleh konsumen, Anda masih perlu mempublikasikan ulang portal Anda. Lihat informasi yang lebih lengkap di [the section called “Publikasikan portal”](#).

Membuat titik akhir REST produk di API Gateway

Titik akhir REST produk adalah titik akses ke produk portal Anda. Setiap titik akhir terdiri dari jalur dan metode REST API dan tahap penerapannya. Titik akhir mencakup dokumentasi apa pun yang telah Anda tetapkan untuk REST API Anda, atau dapat diperbarui dengan dokumentasi khusus. Saat membuat endpoint REST produk, Anda mengonfigurasi pengaturan berikut:

- Nama operasi titik akhir REST produk. Jika REST API Anda memiliki jalur dan metode GET /pets, di portal, namanya adalah /pets. Dengan menggunakan nama operasi, Anda dapat mengubah nama titik akhir REST produk menjadi Pet summaries. Ini tidak memengaruhi API Anda yang ada.
- Fungsi try it untuk titik akhir produk Anda. Ini memungkinkan konsumen API mencoba Anda APIs di portal Anda. Jika Anda membuat titik akhir produk untuk API pribadi, fungsi try it akan disetel ke UNAVAILABLE. Untuk informasi selengkapnya, lihat [the section called “Aktifkan coba untuk titik akhir REST produk di portal Anda”](#).
- [Jika saat ini Anda memiliki bagian dokumentasi yang ditentukan untuk REST API, Anda dapat mengimpor dokumentasi atau mengganti dokumentasi yang ada dan membuat dokumentasi tingkat produk baru menggunakan GitHub Flavored Markdown.](#) Setiap penggantian dokumentasi tidak disebarkan kembali ke bagian dokumentasi API Gateway API Anda. Jika Anda mengimpor bagian dokumentasi saat ini, API Gateway menyinkronkan perubahan di seluruh API Anda ke titik akhir REST produk Anda.

Buat titik akhir REST produk

Prosedur berikut menunjukkan cara menambahkan titik akhir REST produk ke produk portal yang ada. Untuk mempelajari cara membuat produk portal dan titik akhir REST produk bersama-sama, lihat [the section called “Buat produk portal”](#).

Untuk membuat endpoint REST produk untuk portal produk

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk portal.
4. Pilih tab Titik akhir terkait, lalu pilih Tambahkan titik akhir.
5. Untuk memilih titik akhir REST produk Anda, di bawah titik akhir API pilih API, lalu pilih tahapan.
6. Untuk menambahkan titik akhir ke titik akhir REST produk Anda, pilih titik akhir API, lalu pilih Tambahkan ke produk.

Titik akhir API akan muncul di daftar titik akhir API Terpilih.

7. Pilih Kirim.
8. Halaman akhir REST produk Anda sekarang menjadi draf. Pilih halaman referensi Draft API, lalu pilih halaman endpoint REST produk Anda untuk menyelesaikan konten halaman.
9. Pilih Edit halaman.
10. Untuk mengganti dokumentasi API Gateway yang ada, untuk sumber Dokumentasi, aktifkan Ganti dokumentasi dan lakukan hal berikut:
 - a. Untuk Menampilkan konten, pilih Buat penggantian.
 - b. Untuk nama Operasi, masukkan nama operasi baru.
 - c. Untuk badan Halaman, masukkan dokumentasi kustom Anda.
 - d. (Opsional) Untuk Endpoint, masukkan titik akhir kustom yang muncul di portal Anda.
11. Untuk memungkinkan pelanggan menjalankan API Anda di portal Anda, pilih fungsi Coba itu. Untuk informasi selengkapnya, lihat [the section called “Aktifkan coba untuk titik akhir REST produk di portal Anda”](#).
12. Untuk nama Bagian, masukkan nama atau pilih bagian yang ada. Saat Anda menambahkan titik akhir REST produk Anda ke bagian, konsumen dapat melihat konten Anda di portal yang dipublikasikan.
13. Pilih Simpan perubahan.

Agar halaman baru dapat dilihat oleh konsumen, Anda masih perlu mempublikasikan ulang portal Anda. Lihat informasi yang lebih lengkap di [the section called “Publikasikan portal”](#).

Membuat halaman produk di API Gateway

Halaman produk membantu konsumen produk Anda memahami dan menggunakan produk Anda. Dokumentasi berada di tingkat produk, jadi portal apa pun yang menyertakan produk Anda memiliki dokumentasi ini. API Gateway mendukung dua jenis halaman produk yang ditulis dalam [GitHub Flavored Markdown](#):

Halaman Ikhtisar

Saat Anda membuat produk, API Gateway secara otomatis membuat halaman Ikhtisar untuk produk Anda berdasarkan informasi yang Anda berikan saat membuat produk. Halaman ini memiliki format tambahan yang tidak dapat diubah. Anda dapat memodifikasi teks atau menghapus halaman dari produk portal.

Halaman kustom

Halaman kustom adalah halaman yang ditulis dalam Markdown. Anda dapat menggunakan bagian ini untuk syarat dan ketentuan atau tutorial tentang cara menggunakan produk Anda. Anda dapat mengubah halaman kustom menjadi halaman ikhtisar. Saat Anda mengubah halaman kustom menjadi halaman ikhtisar, konten halaman diformat agar sesuai dengan halaman ikhtisar.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan halaman produk Anda:

- Dokumentasi ini tidak berada di tingkat API dan tidak menyertakan metode, sumber daya, atau parameter permintaan apa pun. Untuk mempelajari cara memodifikasi dokumentasi di tingkat API untuk suatu produk, lihat [the section called “Memperbarui titik akhir REST produk”](#)
- Anda tidak dapat langsung mengunggah .md file ke produk Anda.
- Anda dapat mengubah urutan halaman produk dengan memodifikasi daftar halaman dalam urutan tampilan produk portal Anda. Untuk informasi selengkapnya, lihat [the section called “Perbarui produk portal”](#).
- Anda perlu mempublikasikan ulang portal apa pun yang menyertakan produk Anda agar perubahan dokumentasi diterapkan.

- Jika Anda memiliki beberapa portal yang menggunakan produk portal yang sama, halaman produk sama di semua portal. Untuk memiliki titik akhir REST produk yang sama tetapi halaman produk yang berbeda, buat beberapa produk.

Buat halaman produk

Prosedur berikut menunjukkan cara membuat halaman produk.

Untuk membuat halaman produk

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk.
4. Di bawah Dokumentasi, pilih Buat halaman kustom.
5. Untuk judul Halaman, masukkan judul halaman.
6. Untuk Body, masukkan dokumentasi tambahan Anda menggunakan GitHub Flavored Markdown.

Bagian Pratinjau menunjukkan bagaimana konten Anda muncul di portal. Gaya visual akhir mungkin berubah berdasarkan pengaturan portal Anda.

7. Pilih Buat halaman produk.
8. Halaman produk Anda sekarang menjadi draf. Pilih halaman dokumentasi draf, lalu pilih halaman produk Anda untuk menyelesaikan konten halaman.
9. Pilih Edit halaman.
10. Untuk nama Bagian, masukkan nama atau pilih bagian yang ada. Ketika Anda menambahkan halaman produk Anda ke bagian, konsumen dapat melihat konten Anda di portal yang dipublikasikan.
11. Pilih Simpan perubahan.

Agar halaman baru dapat dilihat oleh konsumen, Anda masih perlu mempublikasikan ulang portal Anda. Lihat informasi yang lebih lengkap di [the section called “Publikasikan portal”](#).

Memperbarui produk portal di API Gateway

Ketika Anda memperbarui produk portal, Anda dapat mengubah nama, deskripsi, atau urutan tampilan bagaimana titik akhir REST produk dan halaman produk Anda muncul. Untuk mengubah

urutan tampilan, Anda mengubah daftar halaman untuk halaman produk dan nama bagian untuk titik akhir produk. Jika Anda melakukan ini menggunakan AWS CLI atau SDKs, Anda menentukan halaman produk ARN atau ARN titik akhir REST produk. Anda perlu mempublikasikan ulang portal apa pun yang menggunakan produk Anda untuk konsumen API Anda untuk melihat perubahan yang tercermin dalam portal.

Perbarui produk portal

Prosedur berikut menunjukkan cara mengubah daftar bagian untuk menyusun ulang titik akhir REST produk karena akan muncul di portal. Dalam prosedur ini, kami menganggap Anda memiliki setidaknya dua titik akhir REST produk di produk portal Anda.

Untuk memperbarui produk portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk.
4. Di tab Dokumentasi, untuk halaman Produk, pilih Kelola.
5. Seret dan lepas titik akhir REST produk untuk menyusun ulang mereka. Anda juga dapat memindahkan titik akhir REST produk dan halaman produk ke bagian halaman draf. Halaman di halaman draf tidak akan terlihat di portal Anda.
6. (Opsional) Ganti nama nama titik akhir REST produk atau tambahkan bagian referensi API baru. Perubahan ini tidak akan memengaruhi REST Anda APIs.
7. Pilih Simpan perubahan.

Memperbarui titik akhir REST produk di API Gateway

Saat memperbarui titik akhir REST produk, Anda dapat mengubah semua pengaturan titik akhir produk kecuali jalur dan metode REST API dan tahap penerapannya. Setiap perubahan pada titik akhir produk Anda diperbarui secara otomatis, tetapi Anda perlu mempublikasikan ulang portal apa pun yang menggunakan produk Anda untuk melihat perubahan yang tercermin dalam portal.

Memperbarui titik akhir REST produk

Prosedur berikut menunjukkan cara memperbarui titik akhir REST produk untuk menimpa dokumentasi API saat ini dengan dokumentasi referensi API kustom.

Untuk memperbarui titik akhir REST produk

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk.
4. Di tab Dokumentasi, di bawah halaman referensi API, pilih nama titik akhir REST produk, seperti /dogs - GET.
5. Untuk Pratinjau, pilih Edit halaman.
6. Untuk sumber Dokumentasi, aktifkan Ganti dokumentasi.
7. Untuk Menampilkan konten, pilih Ganti konten yang ada.

Jika Anda memilih Hapus semua konten, konten akan dihapus dari halaman, tetapi Anda dapat memilih Ganti konten yang ada untuk mengakses dokumentasi lagi.

8. Di badan Halaman, masukkan dokumentasi referensi API kustom menggunakan [GitHub Flavored Markdown](#).

Dokumentasi ini tidak akan disebarkan kembali ke API Gateway.

9. Pilih Simpan perubahan.
10. Anda harus menerbitkan ulang portal apa pun yang menggunakan produk portal ini agar perubahan diterapkan.

Memperbarui halaman produk di API Gateway

Saat Anda memperbarui halaman produk, Anda dapat mengubah semua pengaturan halaman produk. Setiap perubahan pada halaman produk Anda diperbarui secara otomatis, tetapi Anda perlu mempublikasikan ulang portal apa pun yang menggunakan produk Anda untuk melihat perubahan yang tercermin dalam portal.

Bagikan produk portal di API Gateway

Anda dapat berbagi produk portal di seluruh Akun AWS penggunaan AWS RAM. Ketika Anda berbagi produk portal, akun lain dapat menggunakan produk portal Anda di portal mereka sendiri. Dengan produk portal bersama, Anda dapat membuat satu katalog organisasi APIs dan menerapkan standar tata kelola di seluruh ekosistem API Anda. Pada saat yang sama, berbagi produk portal memberikan fleksibilitas bagi penyedia API untuk mengembangkan, menguji, dan memelihara APIs di akun mereka sendiri.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi cara Anda berbagi sumber daya portal:

- Saat Anda membagikan produk Anda dengan akun lain, akun tersebut tidak dapat mengubah properti REST API Anda. Ini termasuk titik akhir integrasi, strategi otorisasi, atau konfigurasi tahap.
- Ketika Anda menambahkan produk portal akun lain ke portal Anda, pemilik produk portal tidak dapat melihat atau mengontrol properti lain dari portal Anda. Pemilik produk portal hanya tahu bahwa produk tersebut digunakan di portal Anda.
- Produk portal API Gateway dibagikan di Wilayah AWS level tersebut.
- Anda dapat menggunakan satu pembagian sumber daya dengan beberapa prinsipal, dan setelah Anda membuat pembagian sumber daya, Anda dapat menambahkan lebih banyak prinsipal ke dalamnya. Kami menyarankan bahwa jika memungkinkan, Anda menggunakan kembali bagian sumber daya Anda.
- Jika kedua akun berada di organisasi yang sama menggunakan AWS Organizations, pembagian sumber daya secara otomatis diterima. Anda masih perlu membuat pembagian sumber daya menggunakan AWS RAM.
- Jika kedua akun berada dalam organisasi yang sama menggunakan AWS Organizations dan berbagi sumber daya dalam organisasi Anda diaktifkan, setiap prinsipal di organisasi yang Anda bagikan secara otomatis diberikan akses ke pembagian sumber daya. Tidak perlu undangan dan Anda dapat melewati pembagian sumber daya.
- Jika akun yang Anda bagikan produk tidak menerima pembagian sumber daya dalam waktu 12 jam, Anda harus membagikan sumber daya lagi.
- Setelah Anda membuat pembagian sumber daya, AWS RAM perbarui kebijakan berbagi produk produk Anda untuk mencegah akses ke prinsipal tanpa akses eksplisit. `allow` Untuk informasi selengkapnya, lihat [Menentukan apakah permintaan diizinkan atau ditolak dalam akun](#) di Panduan Pengguna IAM.

Kebijakan sumber daya yang diperbarui akan terlihat seperti berikut:

```
{
  "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 telah mencegah prinsipal tanpa mengizinkan akses eksplisit untuk menambahkan produk Anda ke portal mereka, dengan menambahkan yang berikut:

```
"StringNotEquals": { "aws:PrincipalAccount": "555555555555" }
```

Untuk mempelajari cara berbagi produk, lihat [the section called “Bagikan produk portal Anda dengan pemilik portal”](#). Untuk mempelajari cara menambahkan produk yang dibagikan dengan Anda ke portal Anda, lihat [the section called “Tambahkan produk portal bersama ke portal Anda”](#).

Bagikan produk portal Anda dengan pemilik portal di API Gateway

Sebagai pemilik produk, Anda dapat berbagi produk Anda dengan pemilik portal di Akun AWS penggunaan lain AWS RAM. Anda mempertahankan kontrol penuh atas semua halaman produk dan halaman akhir REST produk.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi cara Anda berbagi produk portal:

- Anda dapat berbagi atau membatalkan berbagi produk portal Anda kapan saja. Jika Anda membatalkan pembagian produk portal saat sedang digunakan di portal yang diterbitkan, pemilik portal tidak akan dapat melihat produk portal, melihat pembaruan apa pun yang dibuat untuk itu, atau menerbitkan ulang atau melihat pratinjau portal hingga mereka menghapus produk portal.
- Anda dapat melihat produk portal yang telah Anda bagikan dengan akun lain.

- Anda tidak dapat memodifikasi portal apa pun yang berisi produk portal Anda, kecuali Anda membuat portal sendiri.

Bagikan produk portal Anda

Prosedur berikut menunjukkan cara membuat berbagi sumber daya.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, lihat [Membuat bagian sumber daya AWS RAM di Panduan AWS RAM Pengguna](#).

Untuk Pilih jenis sumber daya, pilih Produk API Gateway.

AWS CLI

Berikut ini [create-resource-share](#) membuat pembagian sumber daya untuk nama domain kustom pribadi Anda. Diperlukan beberapa menit untuk menyelesaikan sumber daya dan asosiasi utama. Untuk kepala sekolah, berikan ID akun atau ID Organizations, seperti. `arn:aws:organizations::123456789012:organization/o-1234abcd` Anda dapat memberikan beberapa prinsip untuk berbagi sumber daya Anda.

```
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
```

Untuk membatalkan pembagian produk portal Anda, gunakan AWS RAM untuk menghapus pembagian sumber daya.

Kapan saja, Anda dapat memodifikasi kebijakan berbagi produk untuk memodifikasi prinsipal mana yang dapat menggunakan produk portal Anda di portal mereka.

Untuk memperbarui kebijakan berbagi produk

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.

2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk.
4. Di tab Berbagi produk, untuk kebijakan berbagi produk, pilih Kelola kebijakan berbagi produk.
5. Jika Anda belum pernah membagikan produk Anda sebelumnya, kebijakan berbagi produk Anda akan menolak semua akses ke akun lain ke produk portal Anda. Anda perlu memperbarui kebijakan untuk mengizinkan akses akun tertentu ke produk portal Anda.
6. Setelah memperbarui kebijakan berbagi produk, pilih Simpan perubahan.

Menambahkan produk portal bersama ke portal Anda di API Gateway

Sebagai pemilik portal, Anda dapat menggunakan produk portal yang dibagikan dengan Anda oleh akun lain di portal Anda. Anda mempertahankan kontrol penuh atas portal Anda.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi cara Anda berbagi sumber daya portal:

- Anda harus menerima pembagian sumber daya produk dari pemilik produk untuk menggunakan produk mereka di portal Anda. Jika Anda dan pemilik produk berada di Organizations yang sama, AWS RAM dapat menyelesaikan beberapa langkah berbagi untuk Anda.
- Jika pemilik produk tidak membagikan produk saat berada di portal yang dipublikasikan, produk akan tetap terlihat di portal, tetapi Anda tidak akan dapat mempublikasikan portal lagi sampai Anda menghapus produk ini. Adalah tanggung jawab Anda sebagai pemilik portal untuk menghapus produk dari portal Anda, karena pemilik produk tidak dapat melakukannya untuk Anda.
- Anda dapat melihat produk portal, halaman produk, dan halaman titik akhir REST produk, tetapi Anda tidak dapat memodifikasi sumber daya ini.
- Jika suatu produk dibagikan dengan Anda, Anda tidak dapat membagikannya dengan akun lain.

(Opsional) Terima pembagian sumber daya

Setelah pemilik produk Anda membuat pembagian sumber daya, Anda memiliki waktu 12 jam untuk menerimanya. Jika Anda berada di organisasi yang sama menggunakan AWS Organizations sebagai pemilik produk, pembagian secara otomatis diterima. Jika Anda berada di organisasi yang mengaktifkan sumber daya bersama otomatis, sumber daya akan dibagikan secara otomatis dengan Anda.

Konsol Manajemen AWS

Untuk menggunakan Konsol Manajemen AWS, lihat [Menerima dan menolak undangan berbagi sumber daya](#) di Panduan Pengguna AWS RAM

AWS CLI

Untuk menemukan semua sumber daya yang dibagikan dengan Anda, gunakan [get-resource-share-invitations](#) perintah berikut:

```
aws ram get-resource-share-invitations \
  --region us-west-2
```

Gunakan ARN berbagi sumber daya yang dihasilkan untuk menerima undangan berbagi sumber daya. [accept-resource-share-invitation](#) Perintah berikut menerima pembagian sumber daya.

```
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
```

Tambahkan produk bersama ke portal Anda

Setelah Anda menerima pembagian sumber daya, Anda menambahkan produk ke portal Anda.

Konsol Manajemen AWS

Untuk menambahkan produk bersama ke portal Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Di tab Produk, untuk produk Portal, pilih Tambahkan produk.
5. Tambahkan produk ke portal Anda. Produk bersama ditampilkan seperti shared dalam daftar produk.
6. Pilih Simpan perubahan.

AWS CLI

Untuk menemukan semua produk portal yang dibagikan dengan Anda, gunakan `get-portal-products` perintah berikut:

```
aws apigatewayv2 get-portal-products \
  --resource-owner OTHER_ACCOUNTS \
  --region us-west-2
```

Untuk menambahkan produk portal bersama ke portal Anda, gunakan `update-portal` perintah berikut:

```
aws apigateway update-portal \
  --included-portal-product-arns arn:aws:apigateway:us-west-2:111122223333:/
portalproducts/p0000000000 \
  --region us-west-2
```

Aktifkan coba untuk titik akhir REST produk API Gateway di portal Anda

Gunakan coba untuk membiarkan konsumen API memanggil titik akhir produk Anda dari portal Anda. Ketika konsumen API menggunakan mencobanya, mereka memasukkan parameter permintaan metode dan memanggil titik akhir produk Anda. Kemudian, API Gateway memanggil API Anda melalui internet publik dan mengembalikan respons di jendela try it. Anda dapat menggunakan halaman produk khusus untuk memberikan informasi tambahan tentang parameter yang diperlukan untuk mengakses API Anda.

API Gateway menggunakan batasan berikut untuk melindungi Anda APIs:

- API Gateway hanya mengizinkan 3 permintaan per detik ke API Anda.
- API Gateway menggunakan batas waktu tunggu bawaan 29000 ms. API Anda yang sebenarnya mungkin memiliki batas waktu tunggu yang lebih tinggi, tetapi API Gateway tidak menerapkan batas waktu ini saat pelanggan menggunakan mencobanya.
- API Gateway membatasi payload respons hingga 6MB.

Pertimbangan-pertimbangan

Pertimbangan berikut mungkin memengaruhi cara Anda menggunakannya:

- Coba itu tidak didukung saat Anda melihat pratinjau portal.
- Coba itu tidak didukung untuk REST APIs dengan fitur-fitur berikut:
 - Pribadi APIs
 - APIs yang menggunakan TLS timbal balik
 - APIs yang menggunakan sertifikat SSL pribadi atau yang ditandatangani sendiri

Sebagai pemilik portal, Anda bertanggung jawab untuk berkomunikasi dengan konsumen API Anda, alasan mengapa tombol try it tidak ada untuk REST APIs yang tidak didukung. API Gateway tidak menjelaskan hal ini untuk Anda.

Aktifkan coba untuk titik akhir REST produk

Prosedur berikut menunjukkan cara mengaktifkan try it untuk endpoint produk.

Untuk mempelajari cara menggunakannya, coba di portal, lihat [the section called “Gunakan portal”](#).

Aktifkan coba untuk titik akhir REST produk

1. Di panel navigasi utama, pilih produk Portal.
2. Pilih produk.
3. Di tab Dokumentasi, di bawah halaman referensi API, pilih nama titik akhir REST produk, seperti /dogs - GET.
4. Pilih Edit halaman.
5. Untuk memungkinkan pelanggan menjalankan API Anda di portal Anda, pilih Coba fungsionalitas itu.
6. Pilih Simpan perubahan.
7. Anda harus menerbitkan ulang portal apa pun yang menggunakan produk portal ini agar perubahan diterapkan.

Menghapus produk portal di API Gateway

Ketika Anda menghapus produk portal, itu tidak dapat dipulihkan.

Hapus produk portal

Prosedur berikut menunjukkan cara menghapus produk portal.

Untuk menghapus produk portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk portal.
4. Pilih Hapus.
5. Konfirmasikan pilihan Anda lalu pilih Hapus.
6. Anda harus menerbitkan ulang portal apa pun yang menggunakan produk portal ini agar perubahan diterapkan.

Buat portal di API Gateway

Portal adalah kumpulan produk. Anda mengontrol otorisasi, branding, dan publikasi portal Anda. Saat Anda membuat portal, status publikasinya adalah `Disabled`. Ini berarti portal Anda tidak dapat ditemukan di web. Untuk memungkinkan konsumen mengakses portal Anda, Anda perlu mempublikasikannya. Sebelum Anda mempublikasikan portal Anda, Anda juga dapat melihat pratinjaunya.

API Gateway mendukung portal pada browser berikut:

- Firefox
- Google Chrome
- Microsoft Edge
- Safari

Pilih nama domain

Secara default, URL portal Anda di-host di nama domain milik Amazon berikut:

```
https://p-portalId.apigw-portal.us-east-1.on.aws
```

Untuk menyesuaikan nama domain, Anda dapat memberikan nama domain yang Anda miliki dan sertifikat SSL untuk domain kustom. Sertifikat harus dikelola dengan [AWS Certificate Manager](#) (ACM) di AS Timur (Virginia N.) dan didukung oleh Amazon CloudFront. Untuk informasi selengkapnya, lihat [Persyaratan untuk menggunakan SSL/TLS sertifikat dengan CloudFront](#).

Saat Anda menggunakan nama domain khusus, API Gateway membuat CloudFront distribusi, diamankan saat transit dengan sertifikat ACM Anda. Kemudian, API Gateway menyediakan catatan alias, yang Anda tambahkan ke konfigurasi DNS Anda. Catatan alias adalah nama domain milik Amazon. Anda harus menambahkan catatan alias ini untuk mengarahkan lalu lintas ke CloudFront distribusi untuk domain kustom Anda. Nama domain yang Anda berikan harus unik dan belum digunakan oleh daftar alias pada CloudFront distribusi yang ada.

Jika Anda menggunakan nama domain Anda sendiri, API Gateway menggunakan kebijakan TLS_1_2 keamanan untuk melindungi portal Anda. Untuk informasi selengkapnya, lihat [Protokol dan sandi yang didukung](#) antar pemirsa. Anda tidak dapat mengubah nilai ini.

Buat portal

Anda menggunakan konsol API Gateway untuk membuat dan mengonfigurasi portal Anda. Prosedur berikut memandu Anda melalui semua langkah yang diperlukan. Anda juga dapat memilih Skip to Review dan membuat di akhir prosedur pertama untuk membuat portal kosong.

Prosedur berikut menunjukkan cara membuat portal. Pertama, Anda menentukan detail portal Anda.

Untuk menentukan detail portal Anda

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih Buat portal.
4. Di bawah detail portal, untuk nama Portal, masukkan nama.
5. Untuk deskripsi Portal, masukkan deskripsi.
6. Untuk otorisasi Portal, pilih cara mengotorisasi portal Anda.
 - Jika ini adalah pertama kalinya Anda membuat portal, kami sarankan Anda memilih None. Jika Anda berencana menambahkan produksi apa pun APIs ke portal Anda, gunakan Memerlukan otorisasi.
 - Untuk mengharuskan konsumen portal mengautentikasi untuk menggunakan portal Anda, pilih Memerlukan otorisasi, dan lakukan hal berikut:
 - a. Untuk kumpulan pengguna Amazon Cognito, pilih kumpulan pengguna Anda.

Kumpulan pengguna harus berada di Wilayah yang sama dengan portal Anda.
 - b. Untuk ID klien Aplikasi, pilih ID klien aplikasi.

Di klien aplikasi, Anda harus menyetel URL callback ke URL default portal.

- c. Untuk domain kumpulan pengguna, masukkan domain kumpulan pengguna. Untuk informasi selengkapnya, lihat [Mengonfigurasi domain kumpulan pengguna](#).

7. Untuk konfigurasi Domain, pilih nama domain untuk portal Anda.

- Jika ini adalah pertama kalinya Anda membuat portal, kami sarankan Anda memilih Domain default. Setelah Anda mempublikasikan portal Anda, itu akan tersedia di `https://p-portalId.apigw-portal.us-east-1.on.aws`.
- Untuk memberikan nama domain kustom yang Anda miliki, pilih Custom domain name dan lakukan hal berikut:
 - a. Untuk nama Domain, masukkan nama domain yang sepenuhnya memenuhi syarat.
 - b. Untuk sertifikat ACM, pilih sertifikat ACM yang mencakup nama domain.

Anda harus menambahkan catatan alias yang disediakan API Gateway ke catatan DNS Anda, untuk melayani lalu lintas dari portal Anda.

8. Untuk menambahkan monitor aplikasi CloudWatch RUM ke portal Anda, untuk monitor aplikasi RUM, pilih monitor aplikasi. Untuk informasi lebih lanjut, lihat [CloudWatch RUM](#).

9. Pilih Berikutnya.

Setelah Anda menentukan detail portal Anda, Anda menambahkan produk portal. Anda dapat mengubah produk portal di portal Anda kapan saja, tetapi Anda perlu mempublikasikan ulang portal Anda agar perubahan diterapkan.

Untuk memilih produk portal Anda

1. Dari daftar sumber daya, pilih produk yang akan ditambahkan ke portal Anda.
2. Pilih Berikutnya.

Setelah Anda memilih produk Anda, Anda menyesuaikan desain portal. Untuk mengunggah logo portal, Anda harus memiliki gambar yang diunggah ke Amazon S3. Anda dapat mengubah nilai-nilai ini kapan saja, tetapi Anda perlu mempublikasikan ulang portal Anda agar perubahan diterapkan.

(Opsional) Untuk menyesuaikan portal Anda

1. Untuk logo Portal, berikan URI S3 untuk file Anda.

Hanya .svg, .png, atau .jpg didukung.

2. Untuk menyesuaikan desain portal Anda, Anda memiliki dua opsi:
 - a. Anda dapat memilih tema Portal untuk menerapkan tema warna pra-dibuat di portal Anda.
 - b. Anda dapat menggunakan pemilih warna untuk memilih semua opsi desain portal Anda.
3. Pilih Berikutnya.

Setelah Anda memilih desain portal, tinjau informasi untuk portal.

Untuk meninjau portal Anda

- Tinjau pengaturan Anda di halaman ulasan dan buat. Setelah Anda puas dengan pengaturan, pilih Buat.

Jika Anda menggunakan kumpulan pengguna Amazon Cognito untuk mengontrol akses ke portal, Anda harus menyetel URL panggilan balik ke URL default portal di klien aplikasi. Untuk informasi selengkapnya, lihat [Pengaturan khusus aplikasi dengan klien aplikasi](#).

Portal Anda tidak akan tersedia untuk konsumen setelah Anda membuatnya. Anda harus mempublikasikan portal Anda agar dapat diakses di internet. Lihat informasi yang lebih lengkap di [the section called “Publikasikan portal”](#).

Memperbarui portal di API Gateway

Anda dapat memperbarui tema, logo, atau produk portal di portal Anda. Ketika Anda memperbarui portal Anda, perubahan tidak akan berlaku sampai Anda mempublikasikan ulang portal Anda.

Perbarui portal

Prosedur berikut menunjukkan cara memperbarui desain portal.

Untuk memperbarui desain portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Di tab Desain portal, pilih Edit.

5. Untuk memperbarui logo portal, pilih Perbarui logo portal. Anda dapat menghapus logo yang ada atau menggunakan logo baru.
6. Buat pembaruan tema atau latar belakang apa pun untuk desain portal Anda.
7. Pilih Simpan perubahan.
8. (Opsional) Pilih Buat pratinjau portal untuk melihat perubahan Anda.

Setelah mengonfirmasi perubahan, publikasikan portal Anda.

9. Pilih Publikasikan portal.
10. Masukkan deskripsi untuk pembaruan portal Anda dan pilih Publikasikan.

Pratinjau portal di API Gateway

Anda dapat melihat pratinjau portal Anda sebelum mempublikasikannya. API Gateway membuat URL berumur pendek yang dapat Anda gunakan untuk melihat pratinjau versi portal Anda yang akan dilihat pelanggan setelah portal Anda diterbitkan.

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi cara Anda melihat pratinjau portal:

- Coba itu tidak diaktifkan untuk portal pratinjau Anda. Untuk informasi selengkapnya, lihat [the section called “Aktifkan coba untuk titik akhir REST produk di portal Anda”](#).

Pratinjau portal

Prosedur berikut menunjukkan cara melihat pratinjau portal.

Untuk melihat pratinjau portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Pilih Hasilkan pratinjau portal.

Jangan me-refresh halaman Anda.

5. Setelah API Gateway menghasilkan pratinjau Anda, pilih Buka pratinjau portal.

Pratinjau portal Anda akan terbuka di tab baru.

Publikasikan portal di API Gateway

Agar konsumen API dapat mengakses portal Anda, Anda harus mempublikasikannya. URL portal dapat ditemukan oleh siapa saja di internet. Kami menyarankan Anda melihat pratinjau dan mengamankan portal Anda sebelum menerbitkannya.

Pertimbangan-pertimbangan

API Gateway mungkin perlu beberapa menit untuk mempublikasikan portal Anda. Anda dapat memantau status Publikasikan di konsol.

Publikasikan portal

Prosedur berikut menunjukkan cara mempublikasikan portal.

Untuk mempublikasikan portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Pilih Publikasikan portal.
5. (Opsional) Untuk Deskripsi perubahan, masukkan deskripsi perubahan Anda.

Saat Anda mempublikasikan portal, kami sarankan Anda selalu memberikan deskripsi singkat tentang perubahan Anda.

6. Pilih Terbitkan.

API Gateway membutuhkan beberapa menit untuk menyelesaikan penerbitan portal Anda. API Gateway menyediakan tautan ke portal Anda saat tersedia.

Untuk menghapus portal Anda, Anda harus menonaktifkannya terlebih dahulu. Lihat informasi yang lebih lengkap di [the section called “Nonaktifkan portal”](#).

Menggunakan portal API Gateway

Setelah Anda mempublikasikan portal Anda, itu tersedia di web. Untuk menggunakan portal Anda, salin nama domain default Portal atau nama domain dan masukkan di browser. Jika portal Anda menggunakan nama domain, nama domain default tidak dapat digunakan untuk mengakses portal Anda.

Gunakan portal

Prosedur berikut menunjukkan cara mengakses portal dan menggunakannya untuk titik akhir REST produk.

Untuk menggunakan portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Salin nama domain default Portal atau nama domain dan masukkan di browser.
5. Di beranda portal, Anda dapat mencari produk portal dan menemukan halaman produk. Gunakan pencarian di bagian atas portal untuk mencari semua konten di portal Anda. Gunakan bilah pencarian di bawah Produk untuk mencari produk portal.
6. Pilih produk portal.
7. Pilih produk REST endpoint.
8. Di jendela Coba, masukkan parameter untuk mencoba panggilan ke API Anda.
9. Pilih Cobalah. API Gateway akan memanggil API Anda dan menampilkan hasilnya di jendela.

Memecahkan masalah portal Anda

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan portal.

Situs web portal tidak dapat dihubungi

Jika Anda menerima pesan kesalahan bahwa situs web portal tidak dapat dihubungi, pastikan Anda mempublikasikan portal Anda. Jika Anda menggunakan nama domain, pastikan Anda menambahkan

catatan alias yang disediakan API Gateway untuk mengarahkan lalu lintas ke CloudFront distribusi domain kustom Anda.

Mencobanya gagal

Jika Anda tidak dapat menggunakannya untuk titik akhir REST produk, pastikan status try it. ACTIVE Selain itu, pastikan Anda memasukkan parameter yang benar untuk menjalankan API Anda.

Halaman tidak muncul di portal Anda

Jika Anda membuat halaman endpoint REST produk atau halaman produk dan tidak terlihat di portal Anda, pastikan halaman Anda bukan draf. Untuk memverifikasi ini, lakukan hal berikut:

Untuk memverifikasi halaman Anda tidak dalam draf

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih produk Portal.
3. Pilih produk portal Anda.
4. Di tab Dokumentasi, konfirmasi halaman Anda tidak berada di halaman dokumentasi Draft atau bagian halaman referensi API Draft, tetapi berada di halaman dokumentasi kustom atau halaman referensi API.

Nonaktifkan portal di API Gateway

Saat Anda menonaktifkan portal, portal Anda tidak lagi tersedia di web. Setelah Anda menonaktifkan portal, Anda dapat menghapus portal atau menerbitkannya kembali jika perlu.

Nonaktifkan portal

Prosedur berikut menunjukkan cara menonaktifkan portal.

Untuk menonaktifkan portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Pilih Tindakan, lalu pilih Nonaktifkan portal.
5. Konfirmasi pilihan Anda dan pilih Nonaktifkan portal.

Status Publikasikan akan berubah menjadi Dinonaktifkan.

Untuk menghapus status nonaktifkan portal, publikasikan portal lagi.

Menghapus portal di API Gateway

Ketika Anda menghapus portal, itu tidak dapat dipulihkan dan Anda harus menonaktifkannya terlebih dahulu. Jika Anda ingin menghapus portal Anda dari web, Anda dapat menonaktifkannya. Ini memungkinkan Anda memodifikasi portal Anda dan menerbitkannya kembali nanti.

Hapus portal

Prosedur berikut menunjukkan cara menghapus portal.

Untuk menghapus portal

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Di panel navigasi utama, pilih Portal.
3. Pilih portal.
4. Jika portal Anda tidak dinonaktifkan, nonaktifkan dengan memilih Tindakan, Nonaktifkan portal.

Dibutuhkan beberapa menit agar portal Anda dinonaktifkan. Anda dapat memantau status Publikasikan untuk melihat kapan status publikasi portal Anda dinonaktifkan.

5. Setelah Anda tidak mempublikasikan portal Anda, Anda dapat menghapusnya. Untuk menghapus portal Anda, pilih Tindakan, Hapus portal. Konfirmasikan pilihan Anda lalu pilih Hapus.

API Gateway HTTP APIs

REST APIs dan HTTP APIs keduanya adalah produk RESTful API. REST APIs mendukung lebih banyak fitur daripada HTTP APIs, sedangkan HTTP APIs dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. Untuk informasi selengkapnya, lihat [the section called “Pilih antara REST APIs dan HTTP APIs”](#).

Anda dapat menggunakan HTTP APIs untuk mengirim permintaan ke AWS Lambda fungsi atau ke titik akhir HTTP yang dapat dirutekan. Misalnya, Anda dapat membuat API HTTP yang terintegrasi dengan fungsi Lambda di backend. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien.

HTTP APIs mendukung [OpenID Connect dan otorisasi OAuth 2.0](#). Mereka datang dengan dukungan bawaan untuk berbagi sumber daya lintas asal (CORS) dan penerapan otomatis.

Anda dapat membuat HTTP APIs dengan menggunakan AWS Management Console, AWS CLI, APIs, CloudFormation, atau SDKs.

Topik

- [Kembangkan HTTP APIs di API Gateway](#)
- [Publikasikan HTTP APIs agar pelanggan dapat dipanggil](#)
- [Lindungi HTTP Anda APIs di API Gateway](#)
- [Pantau HTTP APIs di API Gateway](#)
- [Memecahkan masalah dengan HTTP APIs di API Gateway](#)

Kembangkan HTTP APIs di API Gateway

Bagian ini memberikan detail tentang kemampuan API Gateway yang Anda butuhkan saat mengembangkan API Gateway APIs.

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Misalnya, Anda mungkin hanya ingin mengizinkan klien tertentu untuk memanggil API Anda, atau Anda mungkin ingin itu tersedia untuk semua orang. Anda mungkin ingin panggilan API untuk menjalankan fungsi Lambda, membuat kueri database, atau memanggil aplikasi.

Topik

- [Buat API HTTP](#)
- [Buat rute untuk HTTP APIs di API Gateway](#)
- [Jenis alamat IP untuk HTTP APIs di API Gateway](#)
- [Kontrol dan kelola akses ke HTTP APIs di API Gateway](#)
- [Buat integrasi untuk HTTP APIs di API Gateway](#)
- [Konfigurasi CORS untuk HTTP APIs di API Gateway](#)
- [Mengubah permintaan dan tanggapan API untuk HTTP APIs di API Gateway](#)
- [Gunakan definisi OpenAPI untuk HTTP APIs di API Gateway](#)

Buat API HTTP

Untuk membuat API fungsional, Anda harus memiliki setidaknya satu rute, integrasi, tahapan, dan penerapan.

Contoh berikut menunjukkan cara membuat API dengan integrasi AWS Lambda atau HTTP, rute, dan tahap default yang dikonfigurasi untuk menerapkan perubahan secara otomatis.

Panduan ini mengasumsikan bahwa Anda sudah terbiasa dengan API Gateway dan Lambda. Untuk panduan yang lebih rinci, lihat [Memulai](#).

Topik

- [Buat API HTTP dengan menggunakan Konsol Manajemen AWS](#)
- [Buat API HTTP dengan menggunakan AWS CLI](#)

Buat API HTTP dengan menggunakan Konsol Manajemen AWS

1. Buka [konsol API Gateway](#).
2. Pilih Buat API.
3. Di bawah HTTP API, pilih Build.
4. Pilih Tambahkan integrasi, lalu pilih AWS Lambda fungsi atau masukkan titik akhir HTTP.
5. Untuk Nama, masukkan nama untuk API Anda.
6. Pilih Periksa dan buat.

7. Pilih Buat.

Sekarang API Anda siap untuk dipanggil. Anda dapat menguji API Anda dengan memasukkan URL pemanggilannya di browser, atau dengan menggunakan Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Buat API HTTP dengan menggunakan AWS CLI

Anda dapat menggunakan quick create untuk membuat API dengan integrasi Lambda atau HTTP, rute catch-all default, dan tahap default yang dikonfigurasi untuk menerapkan perubahan secara otomatis. Perintah [create-api](#) berikut menggunakan quick create untuk membuat API yang terintegrasi dengan fungsi Lambda di backend.

Note

Untuk memanggil integrasi Lambda, API Gateway harus memiliki izin yang diperlukan. Anda dapat menggunakan kebijakan berbasis sumber daya atau IAM role untuk memberikan izin API Gateway guna menjalankan fungsi Lambda. Untuk mempelajari lebih lanjut, lihat [AWS Lambda Izin](#) di Panduan AWS Lambda Pengembang.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target  
arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Sekarang API Anda siap untuk dipanggil. Anda dapat menguji API Anda dengan memasukkan URL pemanggilannya di browser, atau dengan menggunakan Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Buat rute untuk HTTP APIs di API Gateway

Merutekan permintaan API yang masuk langsung ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya—misalnya, `GET /pets`. Anda dapat menentukan metode HTTP spesifik untuk rute Anda. Atau, Anda dapat menggunakan ANY metode untuk mencocokkan semua metode yang belum Anda tetapkan untuk sumber daya. Anda dapat membuat

`$default` rute yang bertindak sebagai tangkapan semua untuk permintaan yang tidak cocok dengan rute lain.

Note

API Gateway menerjemahkan parameter permintaan yang disandikan URL sebelum meneruskannya ke integrasi backend Anda.

Bekerja dengan variabel jalur

Anda dapat menggunakan variabel jalur di rute API HTTP.

Misalnya, GET `/pets/{petID}` rute menangkap GET permintaan yang dikirimkan klien.

`https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`

Variabel jalur serakah menangkap semua sumber daya anak dari suatu rute. Untuk membuat variabel jalur serakah, tambahkan `+` ke nama variabel—misalnya, `{proxy+}`. Variabel jalur serakah harus berada di ujung jalur sumber daya.

Bekerja dengan parameter string kueri

Secara default, API Gateway mengirimkan parameter string kueri ke integrasi backend Anda jika mereka disertakan dalam permintaan ke API HTTP.

Misalnya, ketika klien mengirim permintaan ke `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog`, parameter string kueri `?id=4&type=dog` dikirim ke integrasi Anda.

Bekerja dengan **\$default** rute

`$defaultRoute` menangkap permintaan yang tidak secara eksplisit cocok dengan rute lain di API Anda.

Saat `$default` rute menerima permintaan, API Gateway mengirimkan jalur permintaan lengkap ke integrasi. Misalnya, Anda dapat membuat API hanya dengan `$default` rute dan mengintegrasikannya pada ANY metode dengan titik akhir `https://petstore-demo-endpoint.execute-api.com` HTTP. Saat Anda mengirim permintaan ke `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout`, API Gateway mengirimkan permintaan ke `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Untuk mempelajari lebih lanjut tentang integrasi HTTP, lihat [Buat integrasi proxy HTTP untuk HTTP APIs](#).

Permintaan API perutean

Saat klien mengirim permintaan API, API Gateway terlebih dahulu menentukan [tahap](#) mana yang akan merutekan permintaan tersebut. Jika permintaan secara eksplisit cocok dengan tahapan, API Gateway mengirimkan permintaan ke tahap tersebut. Jika tidak ada tahap yang sepenuhnya cocok dengan permintaan, API Gateway mengirimkan permintaan ke `$default` panggung. Jika tidak ada `$default` tahap, maka API kembali `{"message": "Not Found"}` dan tidak menghasilkan CloudWatch log.

Setelah memilih tahapan, API Gateway memilih rute. API Gateway memilih rute dengan kecocokan paling spesifik, menggunakan prioritas berikut:

1. Pertandingan penuh untuk rute dan metode.
2. Cocokkan rute dan metode dengan variabel jalur serakah (`{proxy+}`).
3. `$defaultRoute`.

Jika tidak ada rute yang cocok dengan permintaan, API Gateway `{"message": "Not Found"}` akan kembali ke klien.

Misalnya, pertimbangkan API dengan `$default` tahapan dan contoh rute berikut:

1. GET `/pets/dog/1`
2. GET `/pets/dog/{id}`
3. GET `/pets/{proxy+}`
4. ANY `/ {proxy+}`
5. `$default`

Tabel berikut merangkum cara API Gateway merutekan permintaan ke rute contoh.

Permintaan	Rute yang dipilih	Penjelasan
GET <code>https://api-id.execute-</code>	GET <code>/pets/dog/1</code>	Permintaan sepenuhnya cocok dengan rute statis ini.

Permintaan	Rute yang dipilih	Penjelasan
api. <i>region</i> .amazonaws.com/pets/dog/1		
GET https://api- <i>id</i> .execute-api. <i>region</i> .amazonaws.com/pets/dog/2	GET /pets/dog/{id}	Permintaan sepenuhnya cocok dengan rute ini.
GET https://api- <i>id</i> .execute-api. <i>region</i> .amazonaws.com/pets/cat/1	GET /pets/{proxy+}	Permintaan tidak sepenuhnya cocok dengan rute. Rute dengan GET metode dan variabel jalur serakah menangkap permintaan ini.
POST https://api- <i>id</i> .execute-api. <i>region</i> .amazonaws.com/test/5	ANY /{proxy+}	ANY Metode ini cocok dengan semua metode yang belum Anda tetapkan untuk rute. Rute dengan variabel jalur serakah memiliki prioritas lebih tinggi daripada \$default rute.

Jenis alamat IP untuk HTTP APIs di API Gateway

Saat membuat API, Anda menentukan jenis alamat IP yang dapat menjalankan API Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk menjalankan API Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat menjalankan API Anda. Anda mungkin ingin mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda:

- Jenis alamat IP default untuk HTTP APIs adalah IPv4.

- Jika Anda mengubah jenis alamat IP untuk API yang ada dari IPv4 ke dualstack, konfirmasi bahwa kebijakan apa pun yang mengontrol akses ke Anda APIs telah diperbarui untuk IPv6 memperhitungkan panggilan. Ketika Anda mengubah jenis alamat IP, perubahan akan segera berlaku.
- API Anda dapat dipetakan ke nama domain khusus dengan jenis alamat IP yang berbeda dari API Anda. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.

Mengubah jenis alamat IP dari API HTTP

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi API. Anda dapat memperbarui konfigurasi API dengan menggunakan Konsol Manajemen AWS, the AWS CLI CloudFormation, atau AWS SDK. Jika Anda mengubah jenis alamat IP API, Anda tidak menerapkan ulang API agar perubahan diterapkan.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP dari API HTTP

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API HTTP.
3. Untuk pengaturan API, pilih Edit.
4. Untuk jenis alamat IP, pilih salah satu IPv4 atau Dualstack.
5. Pilih Simpan.

Perubahan pada konfigurasi API Anda akan segera berlaku.

AWS CLI

Perintah [update-api berikut memperbarui API](#) agar memiliki tipe alamat IP dualstack:

```
aws apigatewayv2 update-api \  
  --api-id abcd1234 \  
  --ip-address-type dualstack
```

Outputnya akan terlihat seperti berikut:

```
{
```

```
"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"
}
```

Kontrol dan kelola akses ke HTTP APIs di API Gateway

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke HTTP API Anda:

- Otorisasi Lambda menggunakan fungsi Lambda untuk mengontrol akses ke. APIs Untuk informasi selengkapnya, lihat [Kontrol akses ke HTTP APIs dengan AWS Lambda otorisasi](#).
- Authorizer JWT menggunakan token web JSON untuk mengontrol akses ke. APIs Untuk informasi selengkapnya, lihat [Kontrol akses ke HTTP APIs dengan otorisasi JWT di API Gateway](#).
- Peran dan kebijakan AWS IAM standar menawarkan kontrol akses yang fleksibel dan kuat. Anda dapat menggunakan peran dan kebijakan IAM untuk mengontrol siapa yang dapat membuat dan mengelola Anda APIs, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat [Kontrol akses ke HTTP APIs dengan otorisasi IAM di API Gateway](#).

Untuk meningkatkan postur keamanan Anda, kami sarankan Anda mengonfigurasi otorisasi untuk semua rute di HTTP API Anda. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Kontrol akses ke HTTP APIs dengan AWS Lambda otorisasi

Anda menggunakan otorisasi Lambda untuk menggunakan fungsi Lambda untuk mengontrol akses ke HTTP API Anda. Kemudian, ketika klien memanggil API Anda, API Gateway memanggil fungsi Lambda Anda. API Gateway menggunakan respons dari fungsi Lambda Anda untuk menentukan apakah klien dapat mengakses API Anda.

Versi format muatan

Versi format payload authorizer menentukan format data yang dikirimkan API Gateway ke pengotorisasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Jika Anda tidak menentukan versi format payload, akan Konsol Manajemen AWS menggunakan versi terbaru secara default. Jika Anda membuat otorisasi Lambda menggunakan AWS CLI, atau SDK CloudFormation, Anda harus menentukan. `authorizerPayloadFormatVersion` Nilai yang di-support adalah `1.0` dan `2.0`.

Jika Anda membutuhkan kompatibilitas dengan REST APIs, gunakan versi `1.0`.

Contoh berikut menunjukkan struktur setiap versi format payload.

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"
}

```

Format respons otorisasi Lambda

Versi format payload juga menentukan struktur respons yang harus Anda kembalikan dari fungsi Lambda Anda.

Respons fungsi Lambda untuk format 1.0

Jika Anda memilih versi 1.0 format, otorisasi Lambda harus menampilkan kebijakan IAM yang mengizinkan atau menolak akses ke rute API Anda. Anda dapat menggunakan sintaks kebijakan IAM standar dalam kebijakan. Untuk contoh kebijakan IAM, lihat [the section called “Kontrol akses untuk menjalankan API”](#). Anda dapat meneruskan properti konteks ke integrasi Lambda atau mengakses log dengan menggunakan `$context.authorizer.property` contextObjek adalah opsional dan `claims` merupakan placeholder yang dicadangkan dan tidak dapat digunakan sebagai objek konteks. Untuk mempelajari selengkapnya, lihat [the section called “Variabel logging”](#).

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"
  }
}
```

Respons fungsi Lambda untuk format 2.0

Jika Anda memilih versi 2.0 format, Anda dapat mengembalikan nilai Boolean atau kebijakan IAM yang menggunakan sintaks kebijakan IAM standar dari fungsi Lambda Anda. Untuk mengembalikan nilai Boolean, aktifkan tanggapan sederhana untuk otorisasi. Contoh berikut menunjukkan format yang harus Anda kodekan fungsi Lambda Anda untuk kembali. contextObjek adalah opsional. Anda dapat meneruskan properti konteks ke integrasi Lambda atau mengakses log dengan

menggunakan. `$context.authorizer.property` Untuk mempelajari selengkapnya, lihat [the section called “Variabel logging”](#).

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"
  }
}
```

Contoh fungsi otorisasi Lambda

Contoh berikut fungsi Lambda Node.js menunjukkan format respons yang diperlukan yang perlu Anda kembalikan dari fungsi Lambda Anda untuk versi format payload. 2.0

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

```

Sumber identitas

Anda dapat secara opsional menentukan sumber identitas untuk otorisasi Lambda. Sumber identitas menentukan lokasi data yang diperlukan untuk mengotorisasi permintaan. Misalnya, Anda dapat menentukan nilai string header atau kueri sebagai sumber identitas. Jika Anda menentukan sumber identitas, klien harus memasukkannya dalam permintaan. Jika permintaan klien tidak menyertakan sumber identitas, API Gateway tidak memanggil otorisasi Lambda Anda, dan klien menerima kesalahan. 401

Tabel berikut menjelaskan sumber identitas yang didukung untuk otorisasi Lambda.

Jenis	Contoh	Catatan
Nilai header	\$ request.header. <i>name</i>	Nama header tidak peka huruf besar/kecil.
Nilai string kueri	\$request.querystring. <i>name</i>	Nama string kueri peka huruf besar/kecil.
Variabel konteks	\$ konteks. <i>variableName</i>	Nilai variabel konteks yang didukung.
Variabel tahap	\$ StageVariables. <i>variableName</i>	Nilai variabel tahap .

Anda juga dapat langsung kembali {"errorMessage" : "Unauthorized"} dari fungsi Lambda Anda untuk mengembalikan 401 kesalahan ke klien Anda. Jika Anda langsung mengembalikan 401 kesalahan dari fungsi Lambda ke klien Anda, jangan tentukan sumber identitas apa pun saat Anda membuat otorisasi Lambda.

Tanggapan otorisasi cache

Anda dapat mengaktifkan caching untuk otorisasi Lambda dengan menentukan file.

[authorizerResultTtlInSeconds](#) Saat caching diaktifkan untuk otorisasi, API Gateway menggunakan sumber identitas otorisasi sebagai kunci cache. Jika klien menentukan parameter yang sama dalam sumber identitas dalam TTL yang dikonfigurasi, API Gateway menggunakan hasil otorisasi yang di-cache, daripada menjalankan fungsi Lambda Anda.

Untuk mengaktifkan caching, otorisasi Anda harus memiliki setidaknya satu sumber identitas.

Jika Anda mengaktifkan respons sederhana untuk otorisasi, respons otorisasi sepenuhnya mengizinkan atau menolak semua permintaan API yang cocok dengan nilai sumber identitas cache. Untuk izin yang lebih terperinci, nonaktifkan tanggapan sederhana dan kembalikan kebijakan IAM. Bergantung pada otorisasi Anda, kebijakan IAM Anda mungkin perlu mengontrol akses ke beberapa.

Secara default, API Gateway menggunakan respons otorisasi yang di-cache untuk semua rute API yang menggunakan otorisasi. Untuk menyimpan respons per rute, tambahkan `$context.routeKey` ke sumber identitas otorisasi Anda.

Buat Authorizer Lambda

Saat membuat otorisasi Lambda, Anda menentukan fungsi Lambda untuk API Gateway yang akan digunakan. Anda harus memberikan izin API Gateway untuk menjalankan fungsi Lambda dengan menggunakan kebijakan sumber daya fungsi atau peran IAM. Perintah [create-authorizer berikut membuat authorizer](#) Lambda:

```
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
```

Perintah [izin tambahan](#) berikut memperbarui kebijakan sumber daya fungsi Lambda untuk memberikan izin API Gateway untuk menjalankan fungsi tersebut. Jika API Gateway tidak memiliki izin untuk menjalankan fungsi Anda, klien akan menerima file. 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"
```

Setelah Anda membuat otorisasi dan memberikan izin API Gateway untuk memanggilnya, perbarui rute Anda untuk menggunakan otorisasi. Perintah [update-route](#) berikut menambahkan Lambda

authorizer ke rute. Jika otorisasi Lambda Anda menggunakan caching kebijakan, pastikan Anda memperbarui kebijakan untuk mengontrol akses untuk rute tambahan.

```
aws apigatewayv2 update-route \  
  --api-id abcdef123 \  
  --route-id abc123 \  
  --authorization-type CUSTOM \  
  --authorizer-id def123
```

Memecahkan masalah otorisasi Lambda

Jika API Gateway tidak dapat memanggil otorisasi Lambda Anda, atau otorisasi Lambda Anda mengembalikan respons dalam format yang tidak valid, klien akan menerima file. 500 Internal Server Error

Untuk memecahkan masalah kesalahan, [aktifkan pencatatan akses untuk tahap](#) API Anda. Sertakan variabel `$context.authorizer.error` logging dalam format log Anda.

Jika log menunjukkan bahwa API Gateway tidak memiliki izin untuk menjalankan fungsi Anda, perbarui kebijakan sumber daya fungsi Anda atau berikan peran IAM untuk memberikan izin API Gateway untuk memanggil otorisasi Anda.

[Jika log menunjukkan bahwa fungsi Lambda Anda mengembalikan respons yang tidak valid, verifikasi bahwa fungsi Lambda Anda mengembalikan respons dalam format yang diperlukan.](#)

Kontrol akses ke HTTP APIs dengan otorisasi JWT di API Gateway

Anda dapat menggunakan JSON Web Tokens (JWTs) sebagai bagian dari [OpenID Connect](#) (OIDC [OAuth](#)) dan 2.0 framework untuk membatasi akses klien ke Anda. APIs

Jika Anda mengonfigurasi otorisasi JWT untuk rute API Anda, API Gateway memvalidasi yang dikirimkan klien dengan JWTs permintaan API. API Gateway memungkinkan atau menolak permintaan berdasarkan validasi token, dan secara opsional, cakupan dalam token. Jika Anda mengonfigurasi cakupan untuk rute, token harus menyertakan setidaknya satu cakupan rute.

Anda dapat mengonfigurasi otorisasi yang berbeda untuk setiap rute API, atau menggunakan otorisasi yang sama untuk beberapa rute.

 Note

Tidak ada mekanisme standar untuk membedakan token akses JWT dari jenis lain JWTs, seperti token ID OpenID Connect. Kecuali Anda memerlukan token ID untuk otorisasi API, kami sarankan Anda mengonfigurasi rute Anda agar memerlukan cakupan otorisasi. Anda juga dapat mengonfigurasi otorisasi JWT Anda untuk meminta penerbit atau audiens yang hanya digunakan penyedia identitas Anda saat mengeluarkan token akses JWT.

Mengotorisasi permintaan API dengan otorisasi JWT

API Gateway menggunakan alur kerja umum berikut untuk mengotorisasi permintaan ke rute yang dikonfigurasi untuk menggunakan otorisasi JWT.

1. [identitySource](#) Periksa token. `identitySource` Dapat hanya menyertakan token, atau token yang diawali dengan `Bearer`.
2. Mendekode token.
3. Periksa algoritma dan tanda tangan token dengan menggunakan kunci publik yang diambil dari penerbit. `jwtks_uri` Saat ini, hanya algoritma berbasis RSA yang didukung. API Gateway dapat menyimpan kunci publik selama dua jam. Sebagai praktik terbaik, saat Anda memutar tombol, izinkan masa tenggang di mana kunci lama dan baru valid.
4. Validasi klaim. API Gateway mengevaluasi klaim token berikut:
 - [kid](#)— Token harus memiliki klaim header yang cocok dengan kunci `jwtks_uri` yang menandatangani token.
 - [iss](#)— Harus cocok dengan [issuer](#) yang dikonfigurasi untuk otorisasi.
 - [aud](#) atau `client_id` — Harus cocok dengan salah satu [audience](#) entri yang dikonfigurasi untuk otorisasi. API Gateway `client_id` hanya memvalidasi jika tidak `aud` ada. Saat keduanya `aud` dan `client_id` ada, API Gateway mengevaluasi. `aud`
 - [exp](#)— Harus setelah waktu saat ini di UTC.
 - [nbf](#)— Harus sebelum waktu saat ini di UTC.
 - [iat](#)— Harus sebelum waktu saat ini di UTC.
 - [scope](#) atau `scp` — Token harus menyertakan setidaknya satu cakupan dalam rute. [authorizationScopes](#)

Jika salah satu langkah ini gagal, API Gateway menolak permintaan API.

Setelah memvalidasi JWT, API Gateway meneruskan klaim dalam token ke integrasi rute API. Sumber daya backend, seperti fungsi Lambda, dapat mengakses klaim JWT. Misalnya, jika JWT menyertakan klaim identitas `emailID`, itu tersedia untuk integrasi Lambda di.

`$event.requestContext.authorizer.jwt.claims.emailID` Untuk informasi selengkapnya tentang payload yang dikirimkan API Gateway ke integrasi Lambda, lihat [the section called “AWS Lambda integrasi”](#)

Buat otorisasi JWT

Sebelum Anda membuat otorisasi JWT, Anda harus mendaftarkan aplikasi klien dengan penyedia identitas. Anda juga harus membuat API HTTP. Untuk contoh pembuatan API HTTP, lihat [Buat API HTTP](#).

Buat otorisasi JWT menggunakan konsol

Langkah-langkah berikut menunjukkan cara membuat otorisasi JWT menggunakan konsol.

Untuk membuat otorisasi JWT menggunakan konsol

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API HTTP.
3. Di panel navigasi utama, pilih Otorisasi.
4. Pilih tab Kelola otorisasi.
5. Pilih Buat.
6. Untuk jenis Authorizer, pilih JWT.
7. Konfigurasi otorisasi JWT Anda, dan tentukan sumber Identitas yang menentukan sumber token.
8. Pilih Buat.

Buat otorisasi JWT menggunakan AWS CLI

Perintah [create-authorizer berikut membuat authorizer](#) JWT. Untuk `jwt-configuration`, tentukan Audience dan Issuer untuk penyedia identitas Anda. Jika Anda menggunakan Amazon Cognito sebagai penyedia identitas, itu adalah. IssuerUrl `https://cognito-idp.us-east-2.amazonaws.com/userPoolID`

```
aws apigatewayv2 create-authorizer \  
  --name authorizer-name \  
  --jwt-configuration jwt-configuration \  
  --identity-provider identity-provider \  
  --scope scope \  
  --stage stage \  
  --api-id api-id \  
  --region region \  
  --output output \  
  --cli-input-json cli-input-json \  
  --profile profile \  
  --role-arn role-arn \  
  --debug
```

```
--api-id api-id \
--authorizer-type JWT \
--identity-source '$request.header.Authorization' \
--jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Buat otorisasi JWT menggunakan AWS CloudFormation

CloudFormation Template berikut membuat API HTTP dengan otorisasi JWT yang menggunakan Amazon Cognito sebagai penyedia identitas.

Output dari CloudFormation template adalah URL untuk UI yang dihosting Amazon Cognito tempat klien dapat mendaftar dan masuk untuk menerima JWT. Setelah klien masuk, klien dialihkan ke HTTP API Anda dengan token akses di URL. Untuk menjalankan API dengan token akses, ubah URL ke a ? untuk menggunakan token sebagai parameter string kueri. #

Contoh CloudFormation template

```
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:
    AllowedOAuthFlows:
      - implicit
    AllowedOAuthScopes:
      - aws.cognito.signin.user.admin
      - email
      - openid
      - profile
    AllowedOAuthFlowsUserPoolClient: 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
```

Perbarui rute untuk menggunakan otorisasi JWT

Anda dapat menggunakan konsol, SDK AWS CLI, atau AWS SDK untuk memperbarui rute untuk menggunakan otorisasi JWT.

Perbarui rute untuk menggunakan otorisasi JWT dengan menggunakan konsol

Langkah-langkah berikut menunjukkan cara memperbarui rute untuk menggunakan otorisasi JWT menggunakan konsol.

Untuk membuat otorisasi JWT menggunakan konsol

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API HTTP.
3. Di panel navigasi utama, pilih Otorisasi.
4. Pilih metode, lalu pilih otorisasi Anda dari menu tarik-turun, dan pilih Lampirkan otorisasi.

Perbarui rute untuk menggunakan otorisasi JWT dengan menggunakan AWS CLI

Perintah [update-route](#) berikut memperbarui rute untuk menggunakan otorisasi JWT:

```
aws apigatewayv2 update-route \
  --api-id api-id \
  --route-id route-id \
  --authorization-type JWT \
  --authorizer-id authorizer-id \
  --authorization-scopes user.email
```

Kontrol akses ke HTTP APIs dengan otorisasi IAM di API Gateway

Anda dapat mengaktifkan otorisasi IAM untuk rute API HTTP. Ketika otorisasi IAM diaktifkan, klien harus menggunakan [Signature Version 4 \(SigV4\)](#) untuk menandatangani permintaan mereka dengan kredensi. AWS API Gateway memanggil rute API Anda hanya jika klien memiliki `execute-api` izin untuk rute tersebut.

[Otorisasi IAM untuk HTTP APIs mirip dengan yang untuk REST. APIs](#)

Note

Kebijakan sumber daya saat ini tidak didukung untuk HTTP APIs.

Untuk contoh kebijakan IAM yang memberikan izin kepada klien untuk memanggil APIs, lihat. [the section called “Kontrol akses untuk menjalankan API”](#)

Aktifkan otorisasi IAM untuk rute

Perintah [update-route](#) berikut memungkinkan otorisasi IAM untuk rute API HTTP:

```
aws apigatewayv2 update-route \  
  --api-id abc123 \  
  --route-id abcdef \  
  --authorization-type AWS_IAM
```

Buat integrasi untuk HTTP APIs di API Gateway

Integrasi menghubungkan rute ke sumber daya backend. HTTP APIs mendukung Lambda proxy, AWS layanan, dan integrasi proxy HTTP. Misalnya, Anda dapat mengonfigurasi POST permintaan ke `/signup` rute API Anda untuk diintegrasikan dengan fungsi Lambda yang menangani pelanggan yang mendaftar.

Topik

- [Buat integrasi AWS Lambda proxy untuk HTTP APIs di API Gateway](#)
- [Buat integrasi proxy HTTP untuk HTTP APIs](#)
- [Buat integrasi AWS layanan untuk HTTP APIs di API Gateway](#)
- [Buat integrasi pribadi untuk HTTP APIs di API Gateway](#)

Buat integrasi AWS Lambda proxy untuk HTTP APIs di API Gateway

Integrasi proxy Lambda memungkinkan Anda mengintegrasikan rute API dengan fungsi Lambda. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien. Untuk contoh pembuatan API HTTP, lihat [Buat API HTTP](#).

Versi format muatan

Versi format payload menentukan format event yang dikirimkan API Gateway ke integrasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Jika Anda tidak menentukan versi format payload, akan Konsol Manajemen AWS menggunakan versi terbaru secara default. Jika Anda membuat integrasi Lambda menggunakan AWS CLI, CloudFormation, atau SDK, Anda harus menentukan `payloadFormatVersion`. Nilai yang di-support adalah `1.0` dan `2.0`.

Untuk informasi selengkapnya tentang cara menyetel `payloadFormatVersion`, lihat [create-integration](#). Untuk informasi selengkapnya tentang cara menentukan integrasi `payloadFormatVersion` yang ada, lihat [get-integration](#).

Perbedaan format payload

Daftar berikut menunjukkan perbedaan antara versi format payload `1.0` dan `2.0` payload:

- Format `2.0` tidak memiliki `multiValueHeaders` atau `multiValueQueryStringParameters` bidang. Header duplikat dikombinasikan dengan koma dan termasuk dalam bidang. `headers` String kueri duplikat digabungkan dengan koma dan disertakan dalam bidang. `queryStringParameters`
- Format `2.0` memiliki `rawPath`. Jika Anda menggunakan pemetaan API untuk menghubungkan stage Anda ke nama domain kustom, tidak `rawPath` akan memberikan nilai pemetaan API. Gunakan format `1.0` dan `path` untuk mengakses pemetaan API untuk nama domain kustom Anda.
- Format `2.0` termasuk `cookies` bidang baru. Semua header cookie dalam permintaan digabungkan dengan koma dan ditambahkan ke bidang. `cookies` Dalam menanggapi klien, setiap cookie menjadi `set-cookie` header.

Struktur format muatan

Contoh berikut menunjukkan struktur setiap versi format payload. Semua headername diberi huruf kecil.

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",
    "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
}

```

Format respons fungsi Lambda

Versi format payload menentukan struktur respons yang harus dikembalikan oleh fungsi Lambda Anda.

Respons fungsi Lambda untuk format 1.0

Dengan versi 1.0 format, integrasi Lambda harus mengembalikan respons dalam format JSON berikut:

Example

```

{
  "isBase64Encoded": true|false,
  "statusCode": httpStatusCode,
  "headers": { "headername": "headervalue", ... },
  "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
  "body": "..."
}

```

```
}
```

Respons fungsi Lambda untuk format 2.0

Dengan versi 2.0 formatnya, API Gateway dapat menyimpulkan format respons untuk Anda. API Gateway membuat asumsi berikut jika fungsi Lambda Anda mengembalikan JSON yang valid dan tidak mengembalikan: statusCode

- isBase64Encoded adalah false.
- statusCode adalah 200.
- content-type adalah application/json.
- body adalah respon dari fungsi tersebut.

Contoh berikut menunjukkan output dari fungsi Lambda dan interpretasi API Gateway.

Keluaran fungsi Lambda	Interpretasi API Gateway
<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>

Untuk menyesuaikan respons, fungsi Lambda Anda harus mengembalikan respons dengan format berikut.

```
{
  "cookies" : ["cookie1", "cookie2"],
  "isBase64Encoded": true|false,
  "statusCode": httpStatusCode,
  "headers": { "headername": "headervalue", ... },
  "body": "Hello from Lambda!"
}
```

Buat integrasi proxy HTTP untuk HTTP APIs

Integrasi proxy HTTP memungkinkan Anda menghubungkan rute API ke titik akhir HTTP yang dapat dirutekan secara publik. Dengan tipe integrasi ini, API Gateway meneruskan seluruh permintaan dan respons antara frontend dan backend.

Untuk membuat integrasi proxy HTTP, berikan URL titik akhir HTTP yang dapat dirutekan secara publik.

Integrasi proxy HTTP dengan variabel jalur

Anda dapat menggunakan variabel jalur di rute API HTTP.

Misalnya, rute `/pets/{petID}` menangkap permintaan ke `/pets/6`. Anda dapat mereferensikan variabel jalur dalam URI integrasi untuk mengirim konten variabel ke integrasi. Contohnya adalah `/pets/extendedpath/{petID}`.

Anda dapat menggunakan variabel jalur serakah untuk menangkap semua sumber daya anak dari suatu rute. Untuk membuat variabel jalur serakah, tambahkan `+` ke nama variabel—misalnya, `{proxy+}`

Untuk menyiapkan rute dengan integrasi proxy HTTP yang menangkap semua permintaan, buat rute API dengan variabel jalur serakah (misalnya, `/parent/{proxy+}`). Integrasikan rute dengan titik akhir HTTP (misalnya, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) pada ANY metode. Variabel jalur serakah harus berada di ujung jalur sumber daya.

Buat integrasi AWS layanan untuk HTTP APIs di API Gateway

Anda dapat mengintegrasikan HTTP API Anda dengan AWS layanan dengan menggunakan integrasi kelas satu. Integrasi kelas satu menghubungkan rute API HTTP ke API AWS layanan. Saat klien

memanggil rute yang didukung oleh integrasi kelas satu, API Gateway akan memanggil API AWS layanan untuk Anda. Misalnya, Anda dapat menggunakan integrasi kelas satu untuk mengirim pesan ke antrian Amazon Simple Queue Service, atau untuk memulai mesin status. AWS Step Functions Untuk tindakan layanan yang didukung, lihat [the section called “AWS referensi integrasi layanan”](#).

Parameter permintaan pemetaan

Integrasi kelas satu memiliki parameter yang diperlukan dan opsional. Anda harus mengkonfigurasi semua parameter yang diperlukan untuk membuat integrasi. Anda dapat menggunakan nilai statis atau parameter peta yang dievaluasi secara dinamis saat runtime. Untuk daftar lengkap integrasi dan parameter yang didukung, lihat [the section called “AWS referensi integrasi layanan”](#).

Tabel berikut menjelaskan parameter permintaan pemetaan yang didukung.

Tipe	Contoh	Catatan
Nilai header	<code>\$ request.header. <i>name</i></code>	Nama header tidak peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai header dengan koma, misalnya <code>"header1": "value1,value2"</code> .
Nilai string kueri	<code>\$request.querystring. <i>name</i></code>	Nama string kueri peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai dengan koma, misalnya <code>"querystring1": "Value1,Value2"</code> .
Parameter jalur	<code>\$request.path. <i>name</i></code>	Nilai parameter jalur dalam permintaan. Misalnya jika rutenya <code>/pets/{petId}</code> , Anda dapat memetakan <code>petId</code> parameter dari permintaan dengan <code><i>\$request.path.petId</i></code> .

Tipe	Contoh	Catatan
Minta passthrough tubuh	\$ request.body	API Gateway melewati seluruh isi permintaan.
Isi permintaan	\$ request.body. <i>name</i>	Ekspresi jalur JSON. Descent rekursif (\$request.body.. <i>name</i>) dan ekspresi filter (?(<i>expression</i>)) tidak didukung.  Note Saat Anda menentukan jalur JSON, API Gateway memotong isi permintaan pada 100 KB dan kemudian menerapkan ekspresi seleksi. Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan \$request.body .
Variabel konteks	\$ konteks. <i>variableName</i>	Nilai variabel konteks yang didukung.
Variabel tahap	\$ StageVariables. <i>variableName</i>	Nilai variabel tahap .
Nilai statis	<i>string</i>	Nilai konstan.

Buat integrasi kelas satu

Sebelum membuat integrasi kelas satu, Anda harus membuat peran IAM yang memberikan izin API Gateway untuk menjalankan tindakan AWS layanan yang Anda integrasikan. Untuk mempelajari lebih lanjut, lihat [Membuat peran untuk AWS layanan](#).

Untuk membuat integrasi kelas satu, pilih tindakan AWS layanan yang didukung, seperti SQS-SendMessage, mengonfigurasi parameter permintaan, dan berikan peran yang memberikan izin API Gateway untuk menjalankan API layanan terintegrasi. AWS Tergantung pada subtype integrasi, parameter permintaan yang berbeda diperlukan. Untuk mempelajari selengkapnya, lihat [the section called “AWS referensi integrasi layanan”](#).

Perintah [create-integration](#) berikut membuat integrasi yang mengirimkan pesan Amazon SQS:

```
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"}'
```

Buat integrasi kelas satu menggunakan CloudFormation

Contoh berikut menunjukkan CloudFormation cuplikan yang membuat `/ {source} / {detailType}` rute dengan integrasi kelas satu dengan Amazon. EventBridge

SourceParameter dipetakan ke parameter `{source}` jalur, DetailType dipetakan ke parameter `{DetailType}` jalur, dan Detail parameter dipetakan ke badan permintaan.

Cuplikan tidak menampilkan bus acara atau peran IAM yang memberikan izin API Gateway untuk menjalankan tindakan. PutEvents

```
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"
```

Referensi subtype integrasi

[Subtype integrasi](#) berikut didukung untuk HTTP APIs.

Subtype integrasi

- [EventBridge- PutEvents 1.0](#)
- [SQS- 1.0 SendMessage](#)
- [SQS- 1.0 ReceiveMessage](#)
- [SQS- 1.0 DeleteMessage](#)
- [SQS- 1.0 PurgeQueue](#)
- [AppConfig- GetConfiguration 1.0](#)
- [Kinesis- 1.0 PutRecord](#)
- [StepFunctions- StartExecution 1.0](#)
- [StepFunctions- StartSyncExecution 1.0](#)
- [StepFunctions- StopExecution 1.0](#)

EventBridge- PutEvents 1.0

Mengirim acara khusus ke Amazon EventBridge sehingga dapat dicocokkan dengan aturan.

Parameter	Diperlukan
Detail	True
DetailType	True
Sumber	True
Waktu	False
EventBusName	False
Sumber daya	False
Wilayah	False
TraceHeader	False

Untuk mempelajari lebih lanjut, lihat [PutEvents](#) di Referensi Amazon EventBridge API.

SQS- 1.0 SendMessage

Mengirimkan pesan ke antrian yang ditentukan.

Parameter	Diperlukan
QueueUrl	True
MessageBody	Benar
DelaySeconds	Salah
MessageAttributes	Salah
MessageDeduplicationId	Salah
MessageGroupId	Salah
MessageSystemAttributes	False
Wilayah	False

Untuk mempelajari selengkapnya, lihat [SendMessage](#) di Referensi API Layanan Antrian Sederhana Amazon.

SQS- 1.0 ReceiveMessage

Mengambil satu atau beberapa pesan (hingga 10), dari antrian yang ditentukan.

Parameter	Diperlukan
QueueUrl	True
AttributeNames	Salah
MaxNumberOfMessages	Salah
MessageAttributeNames	Salah
ReceiveRequestAttemptId	Salah
VisibilityTimeout	Salah
WaitTimeSeconds	False
Wilayah	False

Untuk mempelajari selengkapnya, lihat [ReceiveMessage](#) di Referensi API Layanan Antrian Sederhana Amazon.

SQS- 1.0 DeleteMessage

Menghapus pesan yang ditentukan dari antrian yang ditentukan.

Parameter	Diperlukan
ReceiptHandle	True
QueueUrl	True
Wilayah	False

Untuk mempelajari selengkapnya, lihat [DeleteMessage](#) di Referensi API Layanan Antrian Sederhana Amazon.

SQS- 1.0 PurgeQueue

Menghapus semua pesan dalam antrian yang ditentukan.

Parameter	Diperlukan
QueueUrl	True
Wilayah	False

Untuk mempelajari selengkapnya, lihat [PurgeQueue](#) di Referensi API Layanan Antrian Sederhana Amazon.

AppConfig- GetConfiguration 1.0

Menerima informasi tentang konfigurasi.

Parameter	Diperlukan
Aplikasi	True
Lingkungan	True
Konfigurasi	True
ClientId	Benar
ClientConfigurationVersion	False
Wilayah	False

Untuk mempelajari lebih lanjut, lihat [GetConfiguration](#) di Referensi AWS AppConfig API.

Kinesis- 1.0 PutRecord

Menulis catatan data tunggal ke dalam aliran data Amazon Kinesis.

Parameter	Diperlukan
StreamName	True
Data	True
PartitionKey	Benar
SequenceNumberForOrdering	Salah
ExplicitHashKey	False
Wilayah	False

Untuk mempelajari selengkapnya, lihat [PutRecord](#) di Referensi API Amazon Kinesis Data Streams.

StepFunctions- StartExecution 1.0

Memulai eksekusi mesin negara.

Parameter	Diperlukan
StateMachineArn	True
Nama	False
Input	False
Wilayah	False

Untuk mempelajari lebih lanjut, lihat [StartExecution](#) di Referensi AWS Step Functions API.

StepFunctions- StartSyncExecution 1.0

Memulai eksekusi mesin keadaan sinkron.

Parameter	Diperlukan
StateMachineArn	True

Parameter	Diperlukan
Nama	False
Input	False
Wilayah	False
TraceHeader	False

Untuk mempelajari lebih lanjut, lihat [StartSyncExecution](#) di Referensi AWS Step Functions API.

StepFunctions- StopExecution 1.0

Menghentikan eksekusi.

Parameter	Diperlukan
ExecutionArn	True
Penyebab	False
Kesalahan	False
Wilayah	False

Untuk mempelajari lebih lanjut, lihat [StopExecution](#) di Referensi AWS Step Functions API.

Buat integrasi pribadi untuk HTTP APIs di API Gateway

Integrasi pribadi memungkinkan Anda membuat integrasi API dengan sumber daya pribadi di VPC, seperti Application Load Balancers atau aplikasi berbasis container Amazon ECS.

Anda dapat mengekspos sumber daya Anda dalam VPC untuk diakses oleh klien di luar VPC dengan menggunakan integrasi pribadi. Anda dapat mengontrol akses ke API Anda dengan menggunakan salah satu [metode otorisasi](#) yang didukung API Gateway.

Note

Untuk membuat integrasi pribadi, Anda harus terlebih dahulu membuat tautan VPC. Tautan VPC V2 sekarang didukung untuk HTTP dan REST APIs. Untuk mempelajari selengkapnya tentang tautan VPC V2, lihat [the section called “Tautan VPC V2”](#)

Setelah membuat tautan VPC V2, Anda dapat mengatur integrasi pribadi yang terhubung ke Application Load Balancer, Network Load Balancer, atau sumber daya yang terdaftar dengan layanan. AWS Cloud Map

Pertimbangan-pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan integrasi pribadi Anda:

- Semua sumber daya harus dimiliki oleh yang sama Akun AWS. Ini termasuk penyeimbang beban atau AWS Cloud Map layanan, tautan VPC, dan HTTP API.
- Secara default, lalu lintas integrasi pribadi menggunakan protokol HTTP. Untuk menggunakan HTTPS, tentukan file [tlsConfig](#). Untuk melakukan ini menggunakan Konsol Manajemen AWS, ketika Anda membuat integrasi pribadi Anda, pilih Pengaturan lanjutan dan kemudian masukkan nama server yang aman.
- Untuk integrasi pribadi, API Gateway menyertakan bagian [tahap](#) titik akhir API dalam permintaan ke sumber daya backend Anda. Misalnya, permintaan ke test tahap API termasuk `test/route-path` dalam permintaan untuk integrasi pribadi Anda. Untuk menghapus nama panggung dari permintaan ke sumber daya backend Anda, gunakan [pemetaan parameter](#) untuk menimpa jalur permintaan. `$request.path`

Buat integrasi pribadi menggunakan Application Load Balancer atau Network Load Balancer

Sebelum Anda membuat integrasi pribadi, Anda harus membuat tautan VPC V2. Untuk mempelajari selengkapnya tentang tautan VPC V2, lihat [the section called “Tautan VPC V2”](#)

Untuk membuat integrasi pribadi dengan Application Load Balancer atau Network Load Balancer, buat integrasi proxy HTTP, tentukan tautan VPC yang akan digunakan, dan berikan ARN pendengar penyeimbang beban.

Perintah [create-integration](#) berikut membuat integrasi pribadi yang terhubung ke penyeimbang beban dengan menggunakan tautan VPC:

```
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
```

Buat integrasi pribadi menggunakan penemuan AWS Cloud Map layanan

Sebelum Anda membuat integrasi pribadi, Anda harus membuat tautan VPC V2. Untuk mempelajari lebih lanjut tentang tautan VPC, lihat [the section called “Tautan VPC V2”](#)

Untuk integrasi dengan AWS Cloud Map, API Gateway digunakan `DiscoverInstances` untuk mengidentifikasi sumber daya. Anda dapat menggunakan parameter kueri untuk menargetkan sumber daya tertentu. Atribut sumber daya terdaftar harus menyertakan alamat IP dan port. API Gateway mendistribusikan permintaan di seluruh sumber daya sehat yang dikembalikan. `DiscoverInstances` Untuk mempelajari lebih lanjut, lihat [DiscoverInstances](#) di Referensi AWS Cloud Map API.

Note

Jika Anda menggunakan Amazon ECS untuk mengisi entri AWS Cloud Map, Anda harus mengonfigurasi tugas Amazon ECS untuk menggunakan catatan SRV dengan Amazon ECS Service Discovery atau mengaktifkan Amazon ECS Service Connect. Untuk informasi selengkapnya, lihat [Layanan interkoneksi](#) di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Untuk membuat integrasi pribadi dengan AWS Cloud Map, buat integrasi proxy HTTP, tentukan tautan VPC yang akan digunakan, dan berikan ARN layanan. AWS Cloud Map

Perintah [create-integration](#) berikut menciptakan integrasi pribadi yang menggunakan penemuan AWS Cloud Map layanan untuk mengidentifikasi sumber daya:

```
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
```

Konfigurasi CORS untuk HTTP APIs di API Gateway

[Cross-origin resource sharing \(CORS\)](#) adalah fitur keamanan browser yang membatasi permintaan HTTP yang dimulai dari skrip yang berjalan di browser. Jika Anda tidak dapat mengakses API dan menerima pesan kesalahan yang berisi `Cross-Origin Request Blocked`, Anda mungkin perlu mengaktifkan CORS. Untuk informasi lebih lanjut, lihat [Apa itu CORS?](#).

CORS biasanya diperlukan untuk membangun aplikasi web yang mengakses APIs host pada domain atau asal yang berbeda. Anda dapat mengaktifkan CORS untuk mengizinkan permintaan ke API Anda dari aplikasi web yang dihosting di domain lain. Misalnya, jika API Anda di-host `https://{api_id}.execute-api.{region}.amazonaws.com/` dan Anda ingin memanggil API Anda dari aplikasi web yang di-host `example.com`, API Anda harus mendukung CORS.

Jika Anda mengonfigurasi CORS untuk API, API Gateway secara otomatis mengirimkan respons ke permintaan OPTIONS preflight, meskipun tidak ada rute OPTIONS yang dikonfigurasi untuk API Anda. Untuk permintaan CORS, API Gateway menambahkan header CORS yang dikonfigurasi ke respons dari integrasi.

Note

Jika Anda mengonfigurasi CORS untuk API, API Gateway mengabaikan header CORS yang dikembalikan dari integrasi backend Anda.

Anda dapat menentukan parameter berikut dalam konfigurasi CORS. Untuk menambahkan parameter ini menggunakan API Gateway HTTP API console, pilih Tambah setelah Anda memasukkan nilai.

Header CORS	Properti konfigurasi CORS	Contoh nilai
Access-Control-Allow-Origin	allowOrigins	• <code>https://www.example.com</code> • <code>*</code> (izinkan semua asal) • <code>https://*</code> (izinkan asal apa pun yang dimulai dengan <code>https://</code>)

Header CORS	Properti konfigurasi CORS	Contoh nilai
		<code>http://*</code> (izinkan asal apa pun yang dimulai dengan <code>http://</code>)
Access-Control-Allow-Credentials	<code>allowCredentials</code>	<code>true</code>
Access-Control-Expose-Header	<code>exposeHeaders</code>	<code>tanggal, x-api-id, *</code>
Akses-Kontrol-Max-Age	<code>maxAge</code>	<code>300</code>
Access-Control-Allow-Methods	<code>allowMethods</code>	<code>DAPATKAN, POSTING, HAPUS, *</code>
Access-Control-Allow-Header	<code>allowHeaders</code>	<code>otorisasi, *</code>

Untuk mengembalikan header CORS, permintaan Anda harus berisi header. `origin` Untuk `OPTIONS` metode ini, permintaan Anda harus berisi `origin` header dan `Access-Control-Request-Method` header.

Konfigurasi CORS Anda mungkin terlihat mirip dengan yang berikut ini:

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

https://www.example.com

Add

Access-Control-Allow-Headers

Enter a value for Allowed Headers

authorization

Add

Access-Control-Allow-Methods

Choose Allowed Methods

*

Add

Access-Control-Expose-Headers

Enter a value for Exposed Headers

date,x-api-id

Add

Access-Control-Max-Age

300

Access-Control-Allow-Credentials

☒ YES

Cancel

Save

Mengonfigurasi CORS untuk API HTTP dengan `$default` rute dan otorisasi

Anda dapat mengaktifkan CORS dan mengonfigurasi otorisasi untuk rute apa pun dari API HTTP. Ketika Anda mengaktifkan CORS dan otorisasi untuk `$default` rute, ada beberapa pertimbangan khusus. `$default` rute menangkap permintaan untuk semua metode dan rute yang belum Anda tetapkan secara eksplisit, termasuk permintaan. `OPTIONS` Untuk mendukung `OPTIONS` permintaan yang tidak sah, tambahkan `OPTIONS /{proxy+}` rute ke API Anda yang tidak memerlukan otorisasi dan lampirkan integrasi ke rute. `OPTIONS /{proxy+}` Rute ini memiliki prioritas lebih tinggi daripada `$default` rute. Akibatnya, ini memungkinkan klien untuk mengirimkan `OPTIONS` permintaan ke API Anda tanpa otorisasi. Untuk informasi selengkapnya tentang prioritas perutean, lihat. [Permintaan API perutean](#)

Konfigurasi CORS untuk API HTTP dengan menggunakan CLI AWS

Perintah `update-api` berikut memungkinkan permintaan CORS dari: `https://www.example.com`

CORS

1230

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://www.example.com"
```

Untuk informasi selengkapnya, lihat [CORS](#) di Referensi API Amazon API Gateway Versi 2.

Mengubah permintaan dan tanggapan API untuk HTTP APIs di API Gateway

Anda dapat memodifikasi permintaan API dari klien sebelum mereka mencapai integrasi backend Anda. Anda juga dapat mengubah respons dari integrasi sebelum API Gateway mengembalikan respons ke klien. Anda menggunakan pemetaan parameter untuk memodifikasi permintaan API dan tanggapan untuk HTTP APIs. Untuk menggunakan pemetaan parameter, Anda menentukan permintaan API atau parameter respons untuk dimodifikasi, dan menentukan cara memodifikasi parameter tersebut.

Mengubah permintaan API

Anda menggunakan parameter permintaan untuk mengubah permintaan sebelum mencapai integrasi backend Anda. Anda dapat memodifikasi header, string kueri, atau jalur permintaan.

Parameter permintaan adalah peta nilai kunci. Kunci mengidentifikasi lokasi parameter permintaan untuk diubah, dan bagaimana mengubahnya. Nilai menentukan data baru untuk parameter.

Tabel berikut menunjukkan kunci yang didukung.

Tipe	Sintaks
Header	append overwrite remove:header. <i>headername</i>
String kueri	append overwrite remove:querystring. <i>querystring-name</i>
Jalur	overwrite:path

Tabel berikut menunjukkan nilai yang didukung yang dapat Anda petakan ke parameter.

Tipe	Sintaks	Catatan
Nilai header	<code>\$ request.header. <i>name</i></code> atau <code>{request.header. <i>name</i>}</code>	Nama header tidak peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai header dengan koma, misalnya <code>"header1": "value1,value2"</code> . Beberapa header dicadangkan. Untuk mempelajari selengkapnya, lihat the section called “Header yang dicadangkan” .
Nilai string kueri	<code>\$request.querystring. <i>name</i></code> atau <code>{request.querystring. <i>name</i>}</code>	Nama string kueri peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai dengan koma, misalnya <code>"querystring1" "Value1,Value2"</code> .
Isi permintaan	<code>\$ request.body. <i>name</i></code> atau <code>{request.body. <i>name</i>}</code>	Ekspresi jalur JSON. Penurunan rekursif (<code>\$request.body..name</code>) dan ekspresi filter (<code>(expression)</code>) tidak didukung.

 **Note**
 Saat Anda menentukan jalur JSON, API Gateway memotong isi permintaan pada 100 KB dan kemudian menerapkan ekspresi seleksi.

Tipe	Sintaks	Catatan
		Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan <code>\$request.body</code> .
Jalur permintaan	<code>\$request.path</code> atau <code>\${request.path}</code>	Jalur permintaan, tanpa nama panggung.
Parameter jalur	<code>\$request.path.<i>name</i></code> atau <code>\${request.path.<i>name</i>}</code>	Nilai parameter jalur dalam permintaan. Misalnya jika rutenya <code>/pets/{petId}</code> , Anda dapat memetakan <code>petId</code> parameter dari permintaan dengan <code>\$request.path.petId</code> .
Variabel konteks	<code>\$ konteks.<i>variableName</i></code> atau <code>\${context.<i>variableName</i>}</code>	Nilai variabel konteks .  Note Hanya karakter khusus <code>.</code> dan <code>_</code> didukung.
Variabel tahap	<code>\$ StageVariables.<i>variableName</i></code> atau <code>\${stageVariables.<i>variableName</i>}</code>	Nilai variabel tahap .
Nilai statis	<i>string</i>	Nilai konstan.

 Note

Untuk menggunakan beberapa variabel dalam ekspresi seleksi, lampirkan variabel dalam tanda kurung. Misalnya, `${request.path.name} ${request.path.id}`.

Mengubah respons API

Anda menggunakan parameter respons untuk mengubah respons HTTP dari integrasi backend sebelum mengembalikan respons ke klien. Anda dapat mengubah header atau kode status respons sebelum API Gateway mengembalikan respons ke klien.

Anda mengonfigurasi parameter respons untuk setiap kode status yang dikembalikan oleh integrasi Anda. Parameter respons adalah peta nilai kunci. Kunci mengidentifikasi lokasi parameter permintaan untuk diubah, dan bagaimana mengubahnya. Nilai menentukan data baru untuk parameter.

Tabel berikut menunjukkan kunci yang didukung.

Tipe	Sintaks
Header	<code>append overwrite remove:header. <i>headername</i></code>
Kode status	<code>overwrite:statuscode</code>

Tabel berikut menunjukkan nilai yang didukung yang dapat Anda petakan ke parameter.

Tipe	Sintaks	Catatan
Nilai header	<code>\$ response.header. <i>name</i></code> atau <code>\${response.header. <i>name</i>}</code>	Nama header tidak peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai header dengan koma, misalnya <code>"header1": "value1,value2"</code> . Beberapa header dicadangkan. Untuk mempelajari

Tipe	Sintaks	Catatan
		selengkapnya, lihat the section called “Header yang dicadangkan” .
Isi respons	<code>\$response.body. <i>name</i></code> atau <code>\${response.body. <i>name</i>}</code>	Ekspresi jalur JSON. Descent rekursif (<code>\$response.body..name</code>) dan ekspresi filter (<code>?(expression)</code>) tidak didukung.  Note Saat Anda menentukan jalur JSON, API Gateway memotong badan respons pada 100 KB dan kemudian menerapkan ekspresi pemilihan. Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan <code>\$response.body</code> .
Variabel konteks	<code>\$ konteks. <i>variableName</i></code> atau <code>\${context. <i>variableName</i>}</code>	Nilai variabel konteks yang didukung.
Variabel tahap	<code>\$ StageVariables. <i>variableName</i></code> atau <code>\${stageVariables. <i>variableName</i>}</code>	Nilai variabel tahap .
Nilai statis	<i>string</i>	Nilai konstan.

 Note

Untuk menggunakan beberapa variabel dalam ekspresi seleksi, lampirkan variabel dalam tanda kurung. Misalnya, `${request.path.name} ${request.path.id}`.

Header yang dicadangkan

Header berikut dicadangkan. Anda tidak dapat mengonfigurasi pemetaan permintaan atau respons untuk header ini.

- akses-kontrol-*
- apigw-*
- Otorisasi
- Koneksi
- Pengkodean Konten
- Content-Length
- Content-Location
- Diteruskan
- Tetap Hidup
- Asal
- Proksi-Otentikasi
- Otorisasi Proksi
- TE
- Trailer
- Transfer-Encoding
- Peningkatan
- x-amz-*
- x-amzn-*
- X-Diteruskan-Untuk
- X-Forwarded-Host
- X-Diteruskan-Proto

- Melalui

Contoh

AWS CLI Contoh berikut mengkonfigurasi pemetaan parameter. Misalnya CloudFormation template, lihat [GitHub](#).

Menambahkan header ke permintaan API

Perintah [create-integration](#) berikut membuat header yang diberi nama `header1` ke permintaan API sebelum mencapai integrasi backend Anda. API Gateway mengisi header dengan ID permintaan.

```
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" }'
```

Ganti nama header permintaan

Perintah [create-integration](#) berikut mengganti nama header permintaan dari `header1` ke `header2`

```
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": "" }'
```

Mengubah respon dari integrasi

Perintah [create-integration](#) berikut mengkonfigurasi parameter respons untuk integrasi. Saat integrasi mengembalikan kode status 500, API Gateway mengubah kode status menjadi 403, dan menambahkan `header1 1` ke respons. Saat integrasi mengembalikan kode status 404, API Gateway menambahkan `error header` ke respons.

```
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": "$request.header.header1",  
  "remove:header.header1": "" }'
```

```
--api-id abcdef123 \  
--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"} }'
```

Hapus pemetaan parameter yang dikonfigurasi

Perintah [update-integration](#) berikut menghapus parameter permintaan yang dikonfigurasi sebelumnya untuk `append:header.header1`. Ini juga menghapus parameter respons yang dikonfigurasi sebelumnya untuk kode status 200.

```
aws apigatewayv2 update-integration \  
--api-id abcdef123 \  
--integration-id hijk456 \  
--request-parameters '{"append:header.header1" : ""}' \  
--response-parameters '{"200" : {}}'
```

Gunakan definisi OpenAPI untuk HTTP APIs di API Gateway

Anda dapat menentukan HTTP API Anda dengan menggunakan file definisi OpenAPI 3.0. Kemudian Anda dapat mengimpor definisi ke API Gateway untuk membuat API. Untuk mempelajari lebih lanjut tentang ekstensi API Gateway ke OpenAPI, lihat [Ekstensi OpenAPI](#).

Mengimpor API HTTP

Anda dapat membuat API HTTP dengan mengimpor file definisi OpenAPI 3.0.

Untuk bermigrasi dari REST API ke HTTP API, Anda dapat mengeksport REST API Anda sebagai file definisi OpenAPI 3.0. Kemudian impor definisi API sebagai HTTP API. Untuk mempelajari lebih lanjut tentang mengeksport REST API, lihat [Eksport REST API dari API Gateway](#).

Note

HTTP APIs mendukung AWS variabel yang sama dengan REST APIs. Untuk mempelajari selengkapnya, lihat [AWS variabel untuk impor OpenAPI](#).

Impor informasi validasi

Saat Anda mengimpor API, API Gateway menyediakan tiga kategori informasi validasi.

Info

Properti valid sesuai dengan spesifikasi OpenAPI, tetapi properti itu tidak didukung untuk HTTP APIs.

Misalnya, cuplikan OpenAPI 3.0 berikut menghasilkan info tentang impor karena HTTP APIs tidak mendukung validasi permintaan. API Gateway mengabaikan schema bidang requestBody dan.

```
"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"
        }
      }
    }
  }
}
```

```
    ...  
  }  
  ...  
}
```

Peringatan

Properti atau struktur tidak valid sesuai dengan spesifikasi OpenAPI, tetapi tidak memblokir pembuatan API. Anda dapat menentukan apakah API Gateway harus mengabaikan peringatan ini dan terus membuat API, atau berhenti membuat API pada peringatan.

Dokumen OpenAPI 3.0 berikut menghasilkan peringatan tentang impor karena HTTP APIs hanya mendukung proxy Lambda dan integrasi proxy HTTP.

```
"x-amazon-apigateway-integration": {  
  "type": "AWS",  
  "httpMethod": "POST",  
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",  
  "payloadFormatVersion": "1.0"  
}
```

Kesalahan

Spesifikasi OpenAPI tidak valid atau salah bentuk. API Gateway tidak dapat membuat sumber daya apa pun dari dokumen yang salah format. Anda harus memperbaiki kesalahan, dan kemudian coba lagi.

Definisi API berikut menghasilkan kesalahan pada impor karena HTTP hanya APIs mendukung spesifikasi OpenAPI 3.0.

```
{  
  "swagger": "2.0.0",  
  "info": {  
    "title": "My API",  
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",  
    "version": "1.0"  
  }  
  ...  
}
```

Sebagai contoh lain, sementara OpenAPI memungkinkan pengguna untuk mendefinisikan API dengan beberapa persyaratan keamanan yang dilampirkan pada operasi tertentu, API Gateway

tidak mendukung ini. Setiap operasi hanya dapat memiliki satu otorisasi IAM, otorisasi Lambda, atau otorisasi JWT. Mencoba memodelkan beberapa persyaratan keamanan menghasilkan kesalahan.

Impor API dengan menggunakan AWS CLI

Perintah [import-api](#) berikut mengimpor file `api-definition.json` definisi OpenAPI 3.0 sebagai API HTTP:

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

Anda dapat mengimpor contoh definisi OpenAPI 3.0 berikut untuk membuat API HTTP.

```
{
  "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"  
    ]  
  }  
}  
}
```

Ekspor HTTP APIs dari API Gateway

Setelah membuat API HTTP, Anda dapat mengekspor definisi OpenAPI 3.0 API Anda dari API Gateway. Anda dapat memilih tahap untuk mengekspor, atau mengekspor konfigurasi terbaru API Anda. Anda juga dapat mengimpor definisi API yang diekspor ke API Gateway untuk membuat API lain yang identik. Untuk mempelajari lebih lanjut tentang mengimpor definisi API, lihat [Mengimpor API HTTP](#).

Ekspor definisi OpenAPI 3.0 dari sebuah tahap dengan menggunakan CLI AWS

Perintah [export-api](#) berikut mengekspor definisi OpenAPI dari tahap API yang diberi nama ke file YAML prod bernama `stage-definition.yaml`. File definisi yang diekspor menyertakan [ekstensi API Gateway](#) secara default.

```
aws apigatewayv2 export-api \  
  --api-id api-id \  
  --output-type YAML \  
  --specification OAS30 \  
  --stage-name prod \  
  stage-definition.yaml
```

Ekspor definisi OpenAPI 3.0 dari perubahan terbaru API Anda dengan menggunakan CLI AWS

Perintah [export-api](#) berikut mengekspor definisi OpenAPI dari API HTTP ke file JSON bernama `latest-api-definition.json`. Karena perintah tidak menentukan tahapan, API Gateway mengekspor konfigurasi terbaru API Anda, apakah itu telah diterapkan ke panggung atau belum. File definisi yang diekspor tidak menyertakan [ekstensi API Gateway](#).

```
aws apigatewayv2 export-api \  
  --api-id api-id \  
  latest-api-definition.json
```

```
--output-type JSON \  
--specification OAS30 \  
--no-include-extensions \  
latest-api-definition.json
```

Untuk informasi selengkapnya, lihat [ExportAPI](#) di Referensi API Amazon API Gateway Versi 2.

Ekspor definisi OpenAPI 3.0 menggunakan konsol API Gateway

Prosedur berikut menunjukkan cara mengekspor definisi OpenAPI dari API HTTP.

Untuk mengekspor definisi OpenAPI 3.0 menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API HTTP.
3. Pada panel navigasi utama, di bawah Kembangkan, pilih Ekspor.
4. Pilih dari opsi berikut untuk mengekspor API Anda:

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
\$default ▼

Extensions [Learn more](#) [↗](#)
☒ Include API Gateway extensions

Output format
☒ JSON
☐ YAML

[Download](#)

- a. Untuk Sumber, pilih sumber untuk definisi OpenAPI 3.0. Anda dapat memilih tahap untuk mengekspor, atau mengekspor konfigurasi terbaru API Anda.

- b. Aktifkan Sertakan ekstensi API Gateway untuk menyertakan [ekstensi API Gateway](#).
 - c. Untuk format Output, pilih format output.
5. Pilih Unduh.

Publikasikan HTTP APIs agar pelanggan dapat dipanggil

Anda dapat menggunakan tahapan dan nama domain khusus untuk mempublikasikan API agar klien dapat dipanggil.

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,, dev prodbeta, atauv2). Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil. Anda dapat mengonfigurasi integrasi dan pengaturan yang berbeda untuk setiap tahap API.

Anda dapat menggunakan nama domain khusus untuk memberikan URL yang lebih sederhana dan lebih intuitif bagi klien untuk memanggil API Anda daripada URL default. `https://api-id.execute-api.region.amazonaws.com/stage`

Note

Untuk meningkatkan keamanan API Gateway Anda APIs, `execute-api.region.amazonaws.com` domain terdaftar di [Daftar Akhiran Publik \(PSL\)](#). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway APIs Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman [Set-Cookie](#) di Jaringan Pengembang Mozilla.

Topik

- [Tahapan untuk HTTP APIs di API Gateway](#)
- [Kebijakan keamanan untuk HTTP APIs di API Gateway](#)
- [Nama domain khusus untuk HTTP APIs di API Gateway](#)

Tahapan untuk HTTP APIs di API Gateway

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,, dev prodbeta, atau v2). Tahapan API diidentifikasi oleh ID API dan nama stage mereka, dan mereka disertakan dalam URL yang Anda gunakan untuk memanggil API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

Anda dapat membuat `$default` stage yang disajikan dari dasar URL API Anda—misalnya, `https://{api_id}.execute-api.{region}.amazonaws.com/` Anda menggunakan URL ini untuk memanggil tahap API.

Deployment adalah snapshot dari konfigurasi API Anda. Setelah Anda menerapkan API ke tahap, itu tersedia bagi klien untuk dipanggil. Anda harus menerapkan API agar perubahan diterapkan. Jika Anda mengaktifkan penerapan otomatis, perubahan pada API akan dirilis secara otomatis untuk Anda.

Gunakan variabel tahap untuk HTTP APIs di API Gateway

Variabel tahap adalah pasangan nilai kunci yang dapat Anda tentukan untuk tahap API HTTP. Mereka bertindak seperti variabel lingkungan dan dapat digunakan dalam pengaturan API Anda.

Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensial. Untuk meneruskan data sensitif ke integrasi, gunakan AWS Lambda otorisasi. Anda dapat meneruskan data sensitif ke integrasi dalam output otorisasi Lambda. Untuk mempelajari selengkapnya, lihat [the section called “Format respons otorisasi Lambda”](#).

Contoh - Gunakan variabel panggung untuk menyesuaikan titik akhir integrasi HTTP

Misalnya, Anda dapat menentukan variabel tahap, dan kemudian menetapkan nilainya sebagai titik akhir HTTP untuk integrasi proxy HTTP. Kemudian, Anda dapat mereferensikan titik akhir dengan menggunakan nama variabel tahap terkait. Dengan melakukan ini, Anda dapat menggunakan penyiapan API yang sama dengan titik akhir yang berbeda di setiap tahap. Demikian pula, Anda dapat menggunakan variabel tahap untuk menentukan integrasi AWS Lambda fungsi yang berbeda untuk setiap tahap API Anda.

Untuk menggunakan variabel tahap untuk menyesuaikan titik akhir integrasi HTTP, Anda harus terlebih dahulu mengatur nama dan nilai variabel tahap (misalnya, `url`) dengan `nilaiexample.com`. Selanjutnya, siapkan integrasi proxy HTTP. Alih-alih memasukkan URL titik akhir, Anda dapat memberi tahu API Gateway untuk menggunakan nilai variabel stage, **`http://`**

`${stageVariables.url}`. Nilai ini memberi tahu API Gateway untuk mengganti variabel stage Anda `${}` saat runtime, tergantung pada tahap API Anda.

Anda dapat mereferensikan variabel tahap dengan cara yang sama untuk menentukan nama fungsi Lambda atau peran AWS ARN.

Saat menentukan nama fungsi Lambda sebagai nilai variabel tahap, Anda harus mengonfigurasi izin pada fungsi Lambda secara manual. Perintah [add-permission](#) berikut mengonfigurasi izin untuk fungsi Lambda:

```
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
```

Referensi variabel tahap API Gateway untuk HTTP APIs di API Gateway

Anda dapat menggunakan variabel tahap API Gateway untuk HTTP APIs dalam kasus berikut.

Integrasi HTTP URIs

Anda dapat menggunakan variabel tahap sebagai bagian dari URI integrasi HTTP, seperti yang ditunjukkan pada contoh berikut.

- URI lengkap tanpa protokol — `http://${stageVariables.<variable_name>}`
- Domain lengkap — `http://${stageVariables.<variable_name>}/resource/operation`
- Sebuah subdomain — `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Sebuah jalan — `http://example.com/${stageVariables.<variable_name>}/bar`
- Sebuah string kueri - `http://example.com/foo?q=${stageVariables.<variable_name>}`

Fungsi Lambda

Anda dapat menggunakan variabel tahap sebagai pengganti nama integrasi fungsi Lambda atau alias, seperti yang ditunjukkan pada contoh berikut.

- `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

Untuk menggunakan variabel stage untuk fungsi Lambda, fungsi tersebut harus berada di akun yang sama dengan API. Variabel tahap tidak mendukung fungsi Lambda lintas akun.

AWS kredensi integrasi

Anda dapat menggunakan variabel tahap sebagai bagian dari ARN kredensi AWS pengguna atau peran, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Kebijakan keamanan untuk HTTP APIs di API Gateway

API Gateway memberlakukan kebijakan keamanan TLS_1_2 untuk semua titik akhir HTTP API.

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh Amazon API Gateway. Protokol TLS mengatasi masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server. Ketika klien Anda membuat jabat tangan TLS ke API Anda melalui domain kustom, kebijakan keamanan memberlakukan versi TLS dan pilihan cipher suite yang dapat dipilih klien Anda untuk digunakan. Kebijakan keamanan ini menerima lalu lintas TLS 1.2 dan TLS 1.3 dan menolak lalu lintas TLS 1.0.

Protokol dan cipher TLS yang didukung untuk HTTP APIs

Tabel berikut menjelaskan protokol TLS didukung untuk HTTP. APIs

Protokol TLS	Kebijakan keamanan TLS_1_2
TLSv1.3	Ya
TLSv1.2	Ya

Tabel berikut menjelaskan cipher TLS yang tersedia untuk kebijakan keamanan TLS 1_2 untuk HTTP APIs

Cipher TLS	Kebijakan keamanan TLS_1_2
TLS-AES-128-GCM- SHA256	Ya
TLS-AES-256-GCM- SHA384	Ya
TLS- CHACHA20 - - POLY1305 SHA256	Ya
ECDHE-ECDSA- -GCM- AES128 SHA256	Ya
ECDHE-RSA- -GCM- AES128 SHA256	Ya
ECDHE-ECDSA- - AES128 SHA256	Ya
ECDHE-RSA- - AES128 SHA256	Ya
ECDHE-ECDSA- -GCM- AES256 SHA384	Ya
ECDHE-RSA- -GCM- AES256 SHA384	Ya
ECDHE-ECDSA- - AES256 SHA384	Ya
ECDHE-RSA- - AES256 SHA384	Ya
AES128-GCM- SHA256	Ya
AES128-SHA256	Ya
AES256-GCM- SHA384	Ya

Cipher TLS	Kebijakan keamanan TLS_1_2
AES256-SHA256	Ya

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246 menggunakan nama yang berbeda untuk cipher yang sama. Untuk daftar nama sandi, lihat [the section called “Nama sandi OpenSSL dan RFC”](#)

Informasi tentang REST APIs dan WebSocket APIs

Untuk informasi lebih lanjut tentang REST APIs dan WebSocket APIs, lihat [the section called “Pilih kebijakan keamanan”](#) dan [the section called “Kebijakan keamanan untuk WebSocket APIs di API Gateway”](#).

Nama domain khusus untuk HTTP APIs di API Gateway

Nama domain khusus lebih sederhana dan lebih intuitif URLs yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API Anda, Anda (dan pelanggan Anda) dapat memanggil API menggunakan URL dasar default dari format berikut:

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

dimana *api-id* dihasilkan oleh API Gateway, *region* adalah AWS Region, dan *stage* ditentukan oleh Anda saat menerapkan API.

Bagian nama host dari URL, *api-id*.execute-api.*region*.amazonaws.com mengacu pada titik akhir API. Nama endpoint API default dibuat secara acak, sulit diingat, dan tidak ramah pengguna.

Dengan nama domain khusus, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya, *myservice*) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

Pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan nama domain kustom Anda.

- Nama domain kustom Regional dapat dikaitkan dengan REST APIs dan HTTP APIs. Anda dapat menggunakan API Gateway Versi 2 APIs untuk membuat dan mengelola nama domain kustom Regional untuk REST APIs.
- Untuk versi TLS minimum, hanya TLS 1.2 yang didukung.
- Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS Anda untuk dipetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain khusus tidak dapat mencapai API Gateway.
- Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default dengan menggunakan sertifikat wildcard. Untuk informasi selengkapnya, lihat [the section called “Nama domain kustom wildcard”](#).

Prasyarat

Berikut ini adalah prasyarat untuk membuat nama domain khusus.

Daftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk mengatur nama domain khusus untuk Anda APIs. Anda dapat mendaftarkan nama domain internet Anda menggunakan [Amazon Route 53](#) atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom Anda dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zone apex”) dari domain internet terdaftar.

Nama domain Anda harus mengikuti spesifikasi [RFC 1035](#) dan dapat memiliki maksimum 63 oktet per label dan total 255 oktet.

Sertifikat untuk nama domain kustom

Sebelum menyiapkan nama domain khusus untuk API, Anda harus memiliki sertifikat SSL/TLS yang siap di ACM. Jika ACM tidak tersedia di AWS Wilayah tempat Anda membuat nama domain kustom, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor badan SSL/TLS certificate, you must provide the PEM-formatted SSL/TLS sertifikat, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom.

Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir mengekspos detail sertifikat sensitif apa pun, seperti kunci pribadi. Untuk menggunakan sertifikat AWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

Jika aplikasi Anda menggunakan pinning sertifikat, kadang-kadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelah AWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat [Masalah penyematan sertifikat](#) di Panduan AWS Certificate Manager Pengguna.

Nama domain kustom wildcard

Dengan nama domain khusus wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota [default](#). Misalnya, Anda bisa memberi setiap pelanggan Anda nama domain mereka sendiri `customername.api.example.com`.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (*) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard `*.example.com` menghasilkan subdomain seperti `example.com`, dan `b.example.com`, yang semuanya merutekan ke domain yang sama.

Nama domain kustom wildcard mendukung konfigurasi yang berbeda dari nama domain kustom standar API Gateway. Misalnya, dalam satu AWS akun, Anda dapat mengkonfigurasi `*.example.com` dan `a.example.com` berperilaku berbeda.

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

Note

Anda tidak dapat membuat nama domain khusus wildcard jika AWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuat `example.com`, maka akun B tidak dapat membuat nama `*.example.com` domain khusus wildcard.

Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi [Pusat AWS Dukungan](#) untuk meminta pengecualian.

Langkah selanjutnya untuk nama domain kustom

Untuk menyiapkan nama domain kustom untuk API HTTP, Anda menggunakan dokumentasi dari bagian REST API dari Panduan Pengembang API Gateway.

Pertama, tentukan sertifikat untuk nama domain kustom Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan sertifikat AWS Certificate Manager”](#). Selanjutnya, Anda membuat nama domain kustom Regional. Lihat informasi yang lebih lengkap di [the section called “Siapkan nama domain kustom Regional”](#).

Memetakan tahapan API ke nama domain khusus untuk HTTP APIs

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain khusus. Setelah membuat nama domain dan mengonfigurasi catatan DNS, Anda menggunakan pemetaan API untuk mengirim lalu lintas ke nama domain khusus Anda. APIs

Pemetaan API menentukan API, tahap, dan jalur opsional yang akan digunakan untuk pemetaan. Misalnya, Anda dapat memetakan production tahap API ke `https://api.example.com/orders`.

Anda dapat memetakan tahap HTTP dan REST API ke nama domain kustom yang sama.

Sebelum membuat pemetaan API, Anda harus memiliki API, panggung, dan nama domain khusus. Untuk mempelajari lebih lanjut tentang membuat nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Permintaan API perutean

Anda dapat mengonfigurasi pemetaan API dengan beberapa level, misalnya `orders/v1/items` dan `orders/v2/items`

Untuk pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki jalur pencocokan terpanjang. API Gateway hanya mempertimbangkan jalur yang dikonfigurasi untuk pemetaan API, dan bukan rute API, untuk memilih API yang akan dipanggil. Jika tidak ada jalur yang cocok dengan permintaan, API Gateway mengirimkan permintaan ke API yang telah Anda petakan ke jalur (none) kosong.

Untuk nama domain kustom yang menggunakan pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki awalan pencocokan terpanjang.

Misalnya, pertimbangkan nama domain khusus `https://api.example.com` dengan pemetaan API berikut:

1. `(none)` dipetakan ke API 1.
2. `orders` dipetakan ke API 2.
3. `orders/v1/items` dipetakan ke API 3.
4. `orders/v2/items` dipetakan ke API 4.
5. `orders/v2/items/categories` dipetakan ke API 5.

Permintaan	API yang dipilih	Penjelasan
<code>https://api.example.com/orders</code>	API 2	Permintaan sama persis dengan pemetaan API ini.
<code>https://api.example.com/orders/v1/items</code>	API 3	Permintaan sama persis dengan pemetaan API ini.
<code>https://api.example.com/orders/v2/items</code>	API 4	Permintaan sama persis dengan pemetaan API ini.
<code>https://api.example.com/orders/v1/items/123</code>	API 3	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang g. 123 Pada akhir permintaan tidak mempengaruhi pemilihan .
<code>https://api.example.com/orders/v2/items/categories/5</code>	API 5	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang.
<code>https://api.example.com/customers</code>	API 1	API Gateway menggunakan pemetaan kosong sebagai tangkapan semua.

Permintaan	API yang dipilih	Penjelasan
<code>https://api.example.com/ordersandmore</code>	API 2	API Gateway memilih pemetaan yang memiliki awalan pencocokan terpanjang. Untuk nama domain khusus yang dikonfigurasi dengan pemetaan tingkat tunggal, seperti hanya <code>https://api.example.com/orders</code> dan, API <code>https://api.example.com/</code> Gateway akan memilih API 1, karena tidak ada jalur yang cocok dengannya. <code>ordersandmore</code>

Pembatasan

- Dalam pemetaan API, nama domain khusus dan dipetakan APIs harus berada di akun yang sama AWS .
- Pemetaan API harus hanya berisi huruf, angka, dan karakter berikut: `$-_.+!*'()/`
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda dapat memiliki 200 pemetaan API dengan beberapa level untuk setiap nama domain. Batas ini tidak termasuk pemetaan API dengan level tunggal, seperti `/prod`.
- Anda hanya dapat memetakan HTTP APIs ke nama domain kustom regional dengan kebijakan keamanan TLS 1.2.
- Anda tidak dapat memetakan WebSocket APIs ke nama domain kustom yang sama dengan HTTP API atau REST API.
- Jika Anda membuat pemetaan API dengan beberapa level, API Gateway mengonversi semua nama header menjadi huruf kecil.

Buat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Untuk informasi tentang membuat nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Misalnya AWS Serverless Application Model template yang membuat semua sumber daya, lihat [Sessions With SAM](#) on GitHub.

Konsol Manajemen AWS

Untuk membuat pemetaan API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus.
3. Pilih nama domain khusus yang sudah Anda buat.
4. Pilih pemetaan API.
5. Pilih Konfigurasi pemetaan API.
6. Pilih Tambahkan pemetaan baru.
7. Masukkan API, Stage, dan opsional Path.
8. Pilih Simpan.

AWS CLI

[create-api-mapping](#) Perintah berikut membuat pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan `api.example.com/v1/orders` ke API dan tahap yang ditentukan.

```
aws apigatewayv2 create-api-mapping \  
  --domain-name api.example.com \  
  --api-mapping-key v1/orders \  
  --api-id a1b2c3d4 \  
  --stage test
```

CloudFormation

CloudFormation Contoh berikut membuat pemetaan API.

```
MyApiMapping:  
  Type: 'AWS::ApiGatewayV2::ApiMapping'
```

Properties:

```
DomainName: api.example.com
ApiMappingKey: 'orders/v2/items'
ApiId: !Ref MyApi
Stage: !Ref MyStage
```

Nonaktifkan titik akhir default untuk HTTP APIs

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus, nonaktifkan `execute-api` titik akhir default. Saat Anda menonaktifkan titik akhir default, itu memengaruhi semua tahapan API.

Prosedur berikut menunjukkan cara menonaktifkan endpoint default untuk HTTP API.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih API HTTP.
3. Pilih ID API Anda untuk membuka halaman detail API.
4. Pada detail API, pilih Edit.
5. Untuk titik akhir Default, pilih Nonaktifkan.
6. Pilih Simpan.

Jika Anda mengaktifkan penerapan otomatis untuk tahap Anda, Anda tidak perlu menerapkan ulang API agar perubahan diterapkan. Jika tidak, Anda harus menerapkan ulang API Anda.

7. (Opsional) Pilih Deploy, lalu terapkan ulang API Anda atau buat tahap baru agar perubahan diterapkan.

AWS CLI

[update-domain-name](#) Perintah berikut menonaktifkan titik akhir default untuk HTTP API:

```
aws apigatewayv2 update-api \  
  --api-id abcdef123 \  
  --disable-execute-api-endpoint
```

Setelah menonaktifkan titik akhir default, Anda harus menerapkan API agar perubahan diterapkan, kecuali penerapan otomatis diaktifkan.

Perintah [create-deployment](#) berikut membuat deployment:

```
aws apigatewayv2 create-deployment \  
  --api-id abcdef123 \  
  --stage-name dev
```

Jenis alamat IP untuk nama domain kustom untuk HTTP APIs

Saat membuat API, Anda menentukan jenis alamat IP yang dapat memanggil domain Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk memanggil domain Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat memanggil domain Anda. Kami menyarankan Anda mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda.

- Jenis alamat IP default untuk nama domain kustom API Gateway adalah IPv4.
- Nama domain kustom Anda tidak perlu memiliki jenis alamat IP yang sama untuk semua yang APIs dipetakan ke sana. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.

Mengubah jenis alamat IP dari nama domain kustom

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi endpoint domain. Anda dapat memperbarui konfigurasi endpoint domain dengan menggunakan Konsol Manajemen AWS, the AWS CLI CloudFormation, atau AWS SDK.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP dari nama domain kustom

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama domain kustom publik.

3. Pilih konfigurasi Endpoint.
4. Untuk jenis alamat IP, pilih salah satu IPv4 atau Dualstack.
5. Pilih Simpan.

AWS CLI

[update-domain-name](#) Perintah berikut memperbarui API untuk memiliki tipe alamat IP dualstack:

```
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
```

Outputnya akan terlihat seperti berikut:

```
{
  "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": {}
}
```

Lindungi HTTP Anda APIs di API Gateway

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti pengguna jahat atau lonjakan lalu lintas. Anda dapat melindungi API Anda menggunakan strategi

seperti menyetel target pelambatan, dan mengaktifkan TLS timbal balik. Di bagian ini Anda dapat mempelajari cara mengaktifkan kemampuan ini menggunakan API Gateway.

Topik

- [Permintaan Throttle ke HTTP APIs Anda untuk throughput yang lebih baik di API Gateway](#)
- [Cara mengaktifkan otentikasi TLS timbal balik untuk HTTP APIs di API Gateway](#)

Permintaan Throttle ke HTTP APIs Anda untuk throughput yang lebih baik di API Gateway

Anda dapat mengonfigurasi pembatasan APIs untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Throttle diterapkan atas dasar upaya terbaik dan harus dianggap sebagai target daripada plafon permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritme token bucket, tempat token diperhitungkan untuk permintaan. Secara khusus, API Gateway memeriksa tarif dan ledakan pengiriman permintaan terhadap semua yang ada APIs di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batas dikuasai dalam beberapa kasus.

Jika pengiriman permintaan melebihi tingkat permintaan kondisi tunak dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima tanggapan 429 Too Many Requests kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan atau rute API untuk meningkatkan kinerja keseluruhan APIs di semua akun Anda.

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan kondisi tunak per detik (RPS) APIs di semua AWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua APIs dalam AWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target pengiriman permintaan bersamaan yang akan dipenuhi oleh API Gateway sebelum mengembalikan respons 429 Too Many Requests kesalahan. Untuk informasi lebih lanjut tentang pembatasan kuota, lihat [Kuota dan catatan penting](#)

Batas per akun diterapkan ke semua akun APIs di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan batas waktu APIs yang lebih pendek dan muatan yang lebih kecil. [Untuk meminta peningkatan batas pembatasan tingkat akun per Wilayah, hubungi Pusat Dukungan.AWS](#) Untuk informasi selengkapnya, lihat [Kuota dan catatan penting](#). Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.

Pelambatan tingkat rute

Anda dapat menyetel pembatasan tingkat rute untuk mengganti batas pembatasan permintaan tingkat akun untuk tahap tertentu atau untuk rute individual di API Anda. Batas pembatasan rute default tidak dapat melebihi batas tingkat akun.

Anda dapat mengonfigurasi pelambatan tingkat rute dengan menggunakan file. AWS CLI Perintah [update-stage](#) berikut mengonfigurasi pelambatan kustom untuk tahap dan rute API yang ditentukan:

```
aws apigatewayv2 update-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-settings '{"GET /pets":  
{"ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000}}'
```

Cara mengaktifkan otentikasi TLS timbal balik untuk HTTP Anda APIs di API Gateway

Mutual TLS otentikasi membutuhkan otentikasi dua arah antara klien dan server. Dengan TLS bersama, klien harus menunjukkan sertifikat X.509 untuk memverifikasi identitas mereka untuk mengakses API Anda. Mutual TLS adalah persyaratan umum untuk Internet of Things (IoT) business-to-business dan aplikasi.

Anda dapat menggunakan TLS timbal balik bersama dengan [operasi otorisasi dan otentikasi](#) lain yang didukung API Gateway. API Gateway meneruskan sertifikat yang disediakan klien kepada otorisasi Lambda dan integrasi backend.

Important

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus dengan TLS timbal

balik, nonaktifkan titik `execute-api` akhir default. Untuk mempelajari selengkapnya, lihat [the section called “Nonaktifkan titik akhir default”](#).

Prasyarat untuk TLS timbal balik

Untuk mengkonfigurasi TLS timbal balik yang Anda butuhkan:

- Sebuah nama domain kustom
- Setidaknya satu sertifikat yang dikonfigurasi AWS Certificate Manager untuk nama domain kustom Anda
- Truststore yang dikonfigurasi dan diunggah ke Amazon S3

Nama domain kustom

Untuk mengaktifkan TLS timbal balik untuk API HTTP, Anda harus mengonfigurasi nama domain khusus untuk API Anda. Anda dapat mengaktifkan TLS timbal balik untuk nama domain khusus, dan kemudian memberikan nama domain khusus kepada klien. Untuk mengakses API dengan menggunakan nama domain kustom yang mengaktifkan TLS bersama, klien harus menunjukkan sertifikat yang Anda percayai dalam permintaan API. Anda dapat menemukan informasi lebih lanjut [di the section called “Nama domain kustom”](#).

Menggunakan sertifikat AWS Certificate Manager yang dikeluarkan

Anda dapat meminta sertifikat tepercaya publik langsung dari ACM atau mengimpor sertifikat publik atau yang ditandatangani sendiri. Untuk menyiapkan sertifikat di ACM, buka [ACM](#). Jika Anda ingin mengimpor sertifikat, lanjutkan membaca di bagian berikut.

Menggunakan impor atau AWS Private Certificate Authority sertifikat

Untuk menggunakan sertifikat yang diimpor ke ACM atau sertifikat dari TLS AWS Private Certificate Authority bersama, API Gateway memerlukan yang `ownershipVerificationCertificate` dikeluarkan oleh ACM. Sertifikat kepemilikan ini hanya digunakan untuk memverifikasi bahwa Anda memiliki izin untuk menggunakan nama domain. Ini tidak digunakan untuk jabat tangan TLS. Jika Anda belum memiliki `ownershipVerificationCertificate`, pergi ke <https://console.aws.amazon.com/acm/> untuk mengaturnya.

Anda harus menjaga sertifikat ini tetap berlaku selama masa pakai nama domain Anda. Jika sertifikat kedaluwarsa dan perpanjangan otomatis gagal, semua pembaruan pada nama domain akan

dikunci. Anda perlu memperbarui `ownershipVerificationCertificateArn` dengan valid `ownershipVerificationCertificate` sebelum Anda dapat membuat perubahan lainnya. Ini `ownershipVerificationCertificate` tidak dapat digunakan sebagai sertifikat server untuk domain TLS timbal balik lainnya di API Gateway. Jika sertifikat langsung diimpor kembali ke ACM, penerbit harus tetap sama.

Mengkonfigurasi truststore Anda

Truststores adalah file teks dengan ekstensi `.pem` file. Mereka adalah daftar sertifikat tepercaya dari Otoritas Sertifikat. Untuk menggunakan TLS timbal balik, buat truststore sertifikat X.509 yang Anda percayai untuk mengakses API Anda.

Anda harus menyertakan rantai kepercayaan lengkap, mulai dari sertifikat CA penerbitan, hingga sertifikat CA root, di truststore Anda. API Gateway menerima sertifikat klien yang dikeluarkan oleh CA mana pun yang ada dalam rantai kepercayaan. Sertifikat dapat dari otoritas sertifikat publik atau swasta. Sertifikat dapat memiliki panjang rantai maksimum empat. Anda juga dapat memberikan sertifikat yang ditandatangani sendiri. Algoritma hashing berikut didukung di truststore:

- SHA-256 atau lebih kuat
- RSA-2048 atau lebih kuat
- ECDSA-256 atau lebih kuat

API Gateway memvalidasi sejumlah properti sertifikat. Anda dapat menggunakan otorisasi Lambda untuk melakukan pemeriksaan tambahan saat klien memanggil API, termasuk memeriksa apakah sertifikat telah dicabut. API Gateway memvalidasi properti berikut:

Validasi	Deskripsi
Sintaks X.509	Sertifikat harus memenuhi persyaratan sintaks X.509.
Integritas	Konten sertifikat tidak boleh diubah dari yang ditandatangani oleh otoritas sertifikat dari truststore.
Validitas	Masa berlaku sertifikat harus terkini.

Validasi	Deskripsi
Nama rantai/rantai kunci	Nama dan subjek sertifikat harus membentuk rantai yang tidak terputus. Sertifikat dapat memiliki panjang rantai maksimum empat.

Unggah truststore ke bucket Amazon S3 dalam satu file

Example sertifikat.pem

```
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
<Certificate contents>  
-----END CERTIFICATE-----  
...
```

AWS CLI Perintah [cp](#) berikut diunggah `certificates.pem` ke bucket Amazon S3 Anda:

```
aws s3 cp certificates.pem s3://bucket-name
```

Mengkonfigurasi TLS timbal balik untuk nama domain khusus

Untuk mengonfigurasi TLS timbal balik untuk API HTTP, Anda harus menggunakan nama domain kustom Regional untuk API Anda, dengan versi TLS minimal 1.2. Untuk mempelajari selengkapnya tentang membuat dan mengonfigurasi nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Note

Mutual TLS tidak didukung untuk pribadi APIs.

Setelah Anda mengunggah truststore Anda ke Amazon S3, Anda dapat mengonfigurasi nama domain khusus Anda untuk menggunakan TLS bersama. Berikut ini [create-domain-name](#) membuat nama domain kustom dengan TLS timbal balik:

```
aws apigatewayv2 create-domain-name \  
  --domain-name api.example.com \  
  --domain-name-configurations CertificateArn=arn:aws:acm:us-  
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \  
  --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Setelah membuat nama domain, Anda harus mengonfigurasi catatan DNS dan pemetaan basepath untuk operasi API. Untuk mempelajari selengkapnya, lihat [Siapkan nama domain kustom Regional di API Gateway](#).

Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik

Untuk menjalankan API dengan TLS timbal balik diaktifkan, klien harus menunjukkan sertifikat tepercaya dalam permintaan API. Saat klien mencoba menjalankan API Anda, API Gateway mencari penerbit sertifikat klien di truststore Anda. Agar API Gateway dapat melanjutkan permintaan, penerbit sertifikat dan rantai kepercayaan lengkap hingga sertifikat CA root harus ada di truststore Anda.

Contoh `curl` perintah berikut mengirimkan permintaan ke `api.example.com`, yang termasuk `my-cert.pem` dalam permintaan. `my-key.key` adalah kunci pribadi untuk sertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

API Anda dipanggil hanya jika truststore Anda mempercayai sertifikat. Kondisi berikut akan menyebabkan API Gateway gagal dalam jabat tangan TLS dan menolak permintaan dengan kode 403 status. Jika sertifikat Anda:

- tidak dipercaya
- kedaluwarsa
- tidak menggunakan algoritme yang didukung

Note

API Gateway tidak memverifikasi apakah sertifikat telah dicabut.

Memperbarui truststore Anda

Untuk memperbarui sertifikat di truststore Anda, unggah bundel sertifikat baru ke Amazon S3. Kemudian, Anda dapat memperbarui nama domain kustom Anda untuk menggunakan sertifikat yang diperbarui.

Gunakan versi [Amazon S3 untuk mempertahankan beberapa versi](#) truststore Anda. Saat memperbarui nama domain khusus untuk menggunakan versi truststore baru, API Gateway mengembalikan peringatan jika sertifikat tidak valid.

API Gateway menghasilkan peringatan sertifikat hanya ketika Anda memperbarui nama domain Anda. API Gateway tidak memberi tahu Anda jika sertifikat yang diunggah sebelumnya kedaluwarsa.

[update-domain-name](#) Perintah berikut memperbarui nama domain kustom untuk menggunakan versi truststore baru:

```
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'
```

Nonaktifkan TLS timbal balik

Untuk menonaktifkan TLS timbal balik untuk nama domain kustom, hapus truststore dari nama domain kustom Anda, seperti yang ditunjukkan pada perintah berikut.

[update-domain-name](#) Perintah berikut memperbarui nama domain khusus untuk menghapus truststore dari nama domain kustom Anda:

```
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=''
```

Memecahkan masalah TLS timbal balik untuk HTTP API Anda

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat mengaktifkan TLS timbal balik.

Memecahkan masalah peringatan sertifikat

Saat membuat nama domain khusus dengan TLS timbal balik, API Gateway mengembalikan peringatan jika sertifikat di truststore tidak valid. Ini juga dapat terjadi saat memperbarui nama domain khusus untuk menggunakan truststore baru. Peringatan menunjukkan masalah dengan sertifikat dan subjek sertifikat yang menghasilkan peringatan. Mutual TLS masih diaktifkan untuk API Anda, tetapi beberapa klien mungkin tidak dapat mengakses API Anda.

Anda harus memecahkan kode sertifikat di truststore Anda untuk mengidentifikasi sertifikat mana yang menghasilkan peringatan. Anda dapat menggunakan alat seperti `openssl` untuk memecahkan kode sertifikat dan mengidentifikasi subjek mereka.

Perintah berikut menampilkan isi sertifikat, termasuk subjeknya:

```
openssl x509 -in certificate.crt -text -noout
```

Perbarui atau hapus sertifikat yang menghasilkan peringatan, lalu unggah truststore baru ke Amazon S3. Setelah mengunggah truststore baru, perbarui nama domain kustom Anda untuk menggunakan truststore baru.

Memecahkan masalah konflik nama domain

Kesalahan "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." berarti beberapa Otoritas Sertifikat telah mengeluarkan sertifikat untuk domain ini. Untuk setiap subjek dalam sertifikat, hanya ada satu penerbit di API Gateway untuk domain TLS bersama. Anda harus mendapatkan semua sertifikat Anda untuk subjek itu melalui satu penerbit. Jika masalahnya adalah dengan sertifikat yang tidak Anda kendalikan tetapi Anda dapat membuktikan kepemilikan nama domain, [hubungi Dukungan](#) untuk membuka tiket.

Memecahkan masalah pesan status nama domain

PENDING_CERTIFICATE_REIMPORT: Ini berarti Anda mengimpor ulang sertifikat ke ACM dan validasi gagal karena sertifikat baru memiliki SAN (nama alternatif subjek) yang tidak tercakup oleh `ownershipVerificationCertificate` atau subjek atau SANs dalam sertifikat tidak mencakup nama domain. Sesuatu mungkin dikonfigurasi dengan tidak benar atau sertifikat yang tidak valid diimpor. Anda perlu mengimpor ulang sertifikat yang valid ke ACM. Untuk informasi selengkapnya tentang validasi, lihat [Memvalidasi kepemilikan domain](#).

PENDING_OWNERSHIP_VERIFICATION: Ini berarti sertifikat Anda yang telah diverifikasi sebelumnya telah kedaluwarsa dan ACM tidak dapat memperbaruinya secara otomatis. Anda perlu memperbarui sertifikat atau meminta sertifikat baru. Informasi lebih lanjut tentang perpanjangan sertifikat dapat ditemukan di panduan perpanjangan sertifikat [terkelola pemecahan masalah ACM](#).

Pantau HTTP APIs di API Gateway

Anda dapat menggunakan CloudWatch metrik dan CloudWatch Log untuk memantau HTTP APIs. Dengan menggabungkan log dan metrik, Anda dapat mencatat kesalahan dan memantau kinerja API Anda.

Note

API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:

- 413 Kesalahan Permintaan Entitas Terlalu Besar
- Berlebihan 429 Terlalu Banyak Kesalahan Permintaan
- 400 error seri dari permintaan yang dikirim ke domain kustom yang tidak memiliki pemetaan API
- 500 kesalahan seri yang disebabkan oleh kegagalan internal

Topik

- [Pantau CloudWatch metrik untuk HTTP APIs di API Gateway](#)
- [Konfigurasi logging untuk HTTP APIs di API Gateway](#)

Pantau CloudWatch metrik untuk HTTP APIs di API Gateway

Anda dapat memantau eksekusi API dengan menggunakan CloudWatch, yang mengumpulkan dan memproses data mentah dari API Gateway menjadi metrik yang dapat dibaca. near-real-time Statistik ini dicatat untuk jangka waktu 15 bulan sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, data metrik API Gateway dikirim secara otomatis CloudWatch dalam periode satu menit. Untuk memantau metrik Anda, buat CloudWatch dasbor untuk API Anda. Untuk informasi selengkapnya tentang cara membuat CloudWatch dasbor, lihat [Membuat CloudWatch dasbor](#) di

Panduan CloudWatch Pengguna Amazon. Untuk informasi selengkapnya, lihat [Apa itu Amazon CloudWatch?](#) di Panduan CloudWatch Pengguna Amazon.

Metrik berikut didukung untuk HTTP APIs. Anda juga dapat mengaktifkan metrik terperinci untuk menulis metrik tingkat rute ke Amazon. CloudWatch

Metrik	Deskripsi
4xx	Jumlah kesalahan sisi klien yang ditangkap dalam periode tertentu.
5xx	Jumlah kesalahan sisi server yang ditangkap dalam periode tertentu.
Hitung	Jumlah total permintaan API dalam periode tertentu.
IntegrationLatency	Waktu antara saat API Gateway menyampaikan permintaan ke backend dan saat menerima respons dari backend.
Latensi	Waktu antara saat API Gateway menerima permintaan dari klien dan saat mengembalikan respons ke klien. Latensi mencakup latensi integrasi dan overhead API Gateway lainnya.
DataProcessed	Jumlah data yang diproses dalam byte.

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

Dimensi	Deskripsi
Apild	Memfilter metrik API Gateway untuk API dengan ID API yang ditentukan.
Apild, Panggung	Memfilter metrik API Gateway untuk tahap API dengan ID API dan ID tahap yang ditentukan.
Apild, Metode, Sumber Daya, Panggung	Memfilter metrik API Gateway untuk metode API dengan ID API, ID tahap, jalur sumber daya, dan ID rute yang ditentukan. API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Anda dapat melakukan ini dengan memanggil UpdateStage aksi API Gateway V2 REST API untuk memperbarui <code>detailedMetricsEnabled</code> properti ke <code>true</code>. Atau, Anda dapat memanggil AWS CLI perintah update-stage untuk memperbarui properti ke <code>DetailedMetricsEnabled true</code>. Mengaktifkan metrik tersebut akan dikenakan biaya tambahan ke akun Anda. Untuk informasi

Dimensi	Deskripsi
	harga, lihat CloudWatchHarga Amazon .

Konfigurasikan logging untuk HTTP APIs di API Gateway

Anda dapat mengaktifkan logging untuk menulis log ke CloudWatch Log. Anda dapat menggunakan [variabel logging](#) untuk menyesuaikan konten log Anda.

Untuk meningkatkan postur keamanan Anda, sebaiknya Anda menulis CloudWatch log ke Log untuk semua tahapan API HTTP Anda. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Untuk mengaktifkan logging untuk API HTTP, Anda harus melakukan hal berikut.

1. Pastikan bahwa pengguna Anda memiliki izin yang diperlukan untuk mengaktifkan logging.
2. Buat grup CloudWatch log Log.
3. Berikan ARN dari grup CloudWatch log Log untuk tahap API Anda.

Izin untuk mengaktifkan logging

Untuk mengaktifkan logging untuk API, pengguna Anda harus memiliki izin berikut.

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": "*"
  }
]
```

Buat grup log dan aktifkan logging untuk HTTP APIs

Anda dapat membuat grup log dan mengaktifkan log akses menggunakan Konsol Manajemen AWS atau AWS CLI.

Konsol Manajemen AWS

1. Membuat sebuah grup log.

Untuk mempelajari cara membuat grup log menggunakan konsol, lihat [Membuat Grup Log di Panduan Pengguna CloudWatch Log Amazon](#).

2. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
3. Pilih API HTTP.
4. Di bawah tab Monitor di panel navigasi utama, pilih Logging.
5. Pilih tahap untuk mengaktifkan logging dan pilih Pilih.
6. Pilih Edit untuk mengaktifkan pencatatan akses.
7. Aktifkan Pencatatan akses, masukkan CloudWatch Log, dan pilih format log.

8. Pilih Simpan.

AWS CLI

[create-log-group](#) Perintah berikut membuat grup log:

```
aws logs create-log-group --log-group-name my-log-group
```

Anda memerlukan Nama Sumber Daya Amazon (ARN) untuk grup log Anda untuk mengaktifkan logging. Format ARN adalah `arn:aws:logs::log-group: region account-id log-group-name`

Perintah [update-stage](#) berikut mengaktifkan logging untuk `$default` tahap 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"}'
```

Contoh format log

Contoh beberapa format log akses umum tersedia di konsol API Gateway dan dicantumkan sebagai berikut.

- CLF([Format Log Umum](#)):

```
$context.identity.sourceIp - - [$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(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

Sesuaikan log akses HTTP API

Anda dapat menggunakan variabel berikut untuk menyesuaikan log akses HTTP API. Untuk mempelajari lebih lanjut tentang log akses untuk HTTP APIs, lihat [Konfigurasi logging untuk HTTP APIs di API Gateway](#).

Parameter	Deskripsi
<code>\$context.accountId</code>	ID AWS akun pemilik API.
<code>\$context.apiId</code>	API Gateway identifier ditetapkan ke API Anda.
<code>\$context.authorizer.claims. <i>property</i></code>	Properti klaim yang dikembalikan dari JSON Web Token (JWT) setelah pemanggil metode berhasil diautentikasi, seperti <code>\$context.authorizer.claims.username</code> . Untuk informasi selengkapnya, lihat Kontrol akses ke HTTP APIs dengan otorisasi JWT di API Gateway .

 **Note**
 Memanggil `$context.authorizer.claims` mengembalikan null.

Parameter	Deskripsi
<code>\$context.authorizer.error</code>	Pesan kesalahan dikembalikan dari otorisasi.
<code>\$context.authorizer.</code> <i>property</i>	Nilai pasangan nilai kunci yang ditentukan dari context peta dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan context peta berikut: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> memanggil <code>\$context.authorizer.key</code> mengembalikan "value" string, memanggil <code>\$context.authorizer.numKey</code> mengembalikan 1, dan memanggil <code>\$context.authorizer.boolKey</code> kembali true.
<code>\$context.awsEndpointRequestId</code>	ID permintaan AWS titik akhir dari <code>x-amzn-requestId</code> header <code>x-amz-request-id</code> atau.
<code>\$context.awsEndpointRequestId2</code>	ID permintaan AWS titik akhir dari <code>x-amz-id-2</code> header.
<code>\$context.customDomain.basePathMatched</code>	Jalur untuk pemetaan API yang cocok dengan permintaan masuk. Berlaku ketika klien menggunakan nama domain khusus untuk mengakses API. Misalnya jika klien mengirim permintaan ke <code>https://api.example.com/v1/orders/1234</code> , dan permintaan tersebut cocok dengan pemetaan API dengan jalur <code>v1/orders</code> , nilainya adalah <code>v1/orders</code> . Untuk mempelajari selengkapnya, lihat the section called “Pemetaan API” .

Parameter	Deskripsi
<code>\$context.dataProcessed</code>	Jumlah data yang diproses dalam byte.
<code>\$context.domainName</code>	Nama domain lengkap yang digunakan untuk memanggil API. Ini harus sama dengan Host header yang masuk.
<code>\$context.domainPrefix</code>	Label pertama dari <code>\$context.domainName</code> .
<code>\$context.error.message</code>	String yang berisi pesan kesalahan API Gateway.
<code>\$context.error.messageString</code>	Nilai yang dikutip dari <code>\$context.error.message</code> , yaitu <code>"\$context.error.message"</code> .
<code>\$context.error.responseType</code>	<code>GatewayResponse</code> . Untuk informasi selengkapnya, lihat the section called “Metrik-metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan” .
<code>\$context.extendedRequestId</code>	Setara dengan <code>\$context.requestId</code> .
<code>\$context.httpMethod</code>	Metode HTTP yang digunakan. Nilai yang valid meliputi: DELETE, GET, HEAD, OPTIONS, PATCH, POST, dan PUT.
<code>\$context.identity.accountId</code>	ID AWS akun yang terkait dengan permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.
<code>\$context.identity.caller</code>	Pengidentifikasi utama penelepon yang menandatangani permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.

Parameter	Deskripsi
<code>\$context.identity.cognitoAuthenticationProvider</code>	Daftar dipisahkan koma dari semua penyedia otentikasi Amazon Cognito yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Untuk informasi tentang penyedia autentikasi Amazon Cognito yang tersedia, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Nilai yang mungkin termasuk <code>authenticated</code> untuk identitas yang diautentikasi dan <code>unauthenticated</code> untuk identitas yang tidak diautentikasi.
<code>\$context.identity.cognitoIdentityId</code>	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.

Parameter	Deskripsi
<code>\$context.identity.principalOrgId</code>	ID AWS organisasi . Didukung untuk rute yang menggunakan otorisasi IAM.
<code>\$context.identity.clientCertificate.clientCertPem</code>	Sertifikat klien yang dikodekan PEM yang disajikan klien selama otentikasi TLS timbal balik. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.subjectDN</code>	Nama yang dibedakan dari subjek sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.issuerDN</code>	Nama terkemuka penerbit sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.serialNumber</code>	Nomor seri sertifikat. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.validity.notBefore</code>	Tanggal sebelum sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	Tanggal setelah sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.

Parameter	Deskripsi
<code>\$context.identity.sourceIp</code>	Alamat IP sumber dari koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.user</code>	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk rute yang menggunakan otorisasi IAM.
<code>\$context.identity.userAgent</code>	User-Agent Header pemanggil API.
<code>\$context.identity.userArn</code>	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi. Didukung untuk rute yang menggunakan otorisasi IAM. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.integration.error</code>	Pesan kesalahan dikembalikan dari integrasi. Setara dengan <code>\$context.integration.errorMessage</code> .
<code>\$context.integration.integrationStatus</code>	Untuk integrasi proxy Lambda, kode status dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
<code>\$context.integration.latency</code>	Latensi integrasi dalam ms. Setara dengan <code>\$context.integrationLatency</code> .
<code>\$context.integration.requestId</code>	ID permintaan AWS titik akhir. Setara dengan <code>\$context.awsEndpointRequestId</code> .

Parameter	Deskripsi
<code>\$context.integration.status</code>	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda.
<code>\$context.integrationErrorMessage</code>	String yang berisi pesan kesalahan integrasi.
<code>\$context.integrationLatency</code>	Latensi integrasi dalam ms.
<code>\$context.integrationStatus</code>	Untuk integrasi proxy Lambda, parameter ini mewakili kode status yang dikembalikan dari AWS Lambda, bukan dari fungsi Lambda backend.
<code>\$context.path</code>	Jalur permintaan. Misalnya, <code>/[stage]/root/child</code> .
<code>\$context.protocol</code>	Protokol permintaan, misalnya, <code>HTTP/1.1</code> .
 Note API Gateway APIs dapat menerima permintaan HTTP/2, tetapi API Gateway mengirimkan permintaan ke integrasi backend menggunakan HTTP/1.1. Akibatnya, protokol permintaan dicatat sebagai HTTP/1.1 bahkan jika klien mengirim permintaan yang menggunakan HTTP/2.	
<code>\$context.requestId</code>	ID yang ditetapkan API Gateway ke permintaan API.
<code>\$context.requestTime</code>	Waktu permintaan yang diformat CLF (). <code>dd/MM/yyyy:HH:mm:ss +-hhmm</code>

Parameter	Deskripsi
<code>\$context.requestTimeEpoch</code>	Waktu permintaan yang diformat Epoch .
<code>\$context.responseLatency</code>	Latensi respons dalam ms.
<code>\$context.responseLength</code>	Panjang payload respon dalam byte.
<code>\$context.routeKey</code>	Kunci rute permintaan API, misalnya/pets.
<code>\$context.stage</code>	Tahap penerapan permintaan API (misalnya, beta atau prod).
<code>\$context.status</code>	Status respons metode.

Memecahkan masalah dengan HTTP APIs di API Gateway

Topik berikut memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan HTTP APIs

Topik

- [Memecahkan masalah dengan integrasi Lambda API HTTP](#)
- [Memecahkan masalah dengan otorisasi HTTP API JWT](#)

Memecahkan masalah dengan integrasi Lambda API HTTP

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan [AWS Lambda integrasi](#) dengan HTTP APIs

Masalah: API saya dengan integrasi Lambda kembali **{"message": "Internal Server Error"}**

Untuk memecahkan masalah kesalahan server internal, tambahkan [variabel `\$context.integrationErrorMessage` logging](#) ke format log Anda, dan lihat log HTTP API Anda. Untuk mencapai ini, lakukan hal berikut:

Untuk membuat grup log dengan menggunakan Konsol Manajemen AWS

1. Buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Grup log.
3. Pilih Buat grup log.
4. Masukkan nama grup log, lalu pilih Buat.
5. Perhatikan Nama Sumber Daya Amazon (ARN) untuk grup log Anda. Format ARN adalah `arn:aws:logs::log-group: region account-id log-group-name` Anda memerlukan grup log ARN untuk mengaktifkan pencatatan akses untuk HTTP API Anda.

Untuk menambahkan variabel `$context.integrationErrorMessage` logging

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih HTTP API Anda.
3. Di bawah Monitor, pilih Logging.
4. Pilih tahap API Anda.
5. Pilih Edit, lalu aktifkan pencatatan akses.
6. Untuk tujuan Log, masukkan ARN dari grup log yang Anda buat pada langkah sebelumnya.
7. Untuk format Log, pilih CLF. API Gateway membuat contoh format log.
8. Tambahkan `$context.integrationErrorMessage` ke akhir format log.
9. Pilih Simpan.

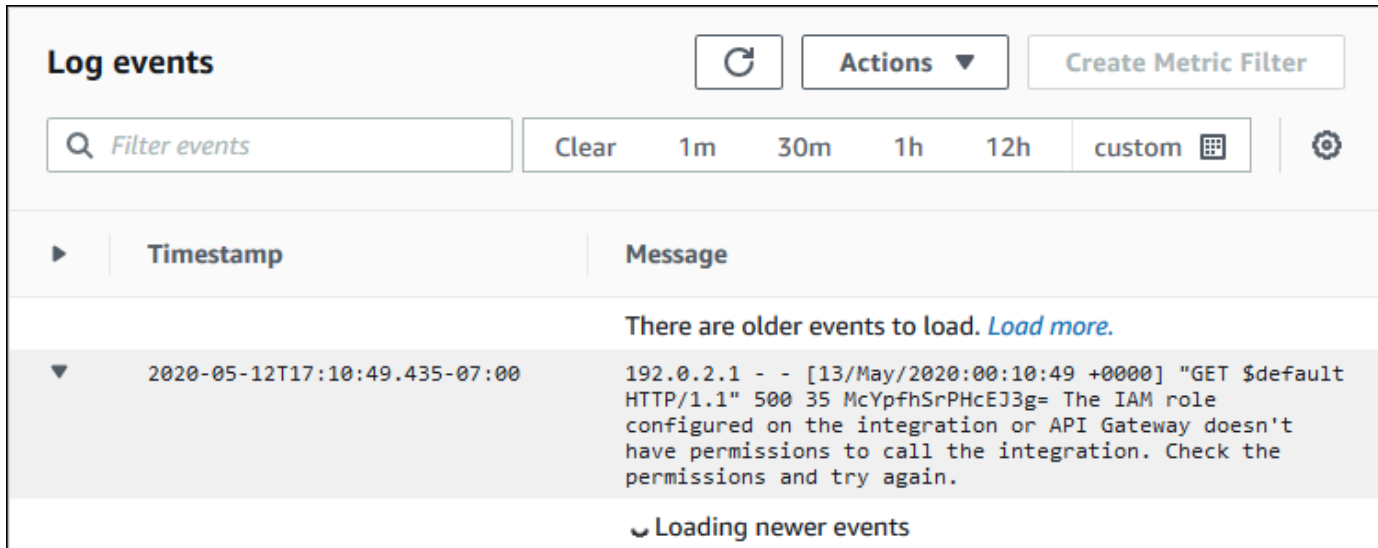
Untuk melihat log API Anda

1. Hasilkan log. Gunakan browser atau `curl` untuk menjalankan API Anda.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
3. Pilih HTTP API Anda.
4. Di bawah Monitor, pilih Logging.
5. Pilih tahap API yang Anda aktifkan logging.
6. Pilih Lihat log masuk CloudWatch.
7. Pilih aliran log terbaru untuk melihat log HTTP API Anda.

8. Entri log Anda akan terlihat mirip dengan yang berikut ini:



Karena kami menambahkan `$context.integrationErrorMessage` ke format log, kami melihat pesan kesalahan di log kami yang merangkum masalah.

Log Anda mungkin menyertakan pesan kesalahan berbeda yang menunjukkan bahwa ada masalah dengan kode fungsi Lambda Anda. [Dalam hal ini, periksa kode fungsi Lambda Anda, dan verifikasi bahwa fungsi Lambda Anda mengembalikan respons dalam format yang diperlukan.](#)

Jika log Anda tidak menyertakan pesan kesalahan, tambahkan `$context.error.message` dan `$context.error.responseType` ke format log Anda untuk informasi selengkapnya guna membantu memecahkan masalah.

Dalam kasus ini, log menunjukkan bahwa API Gateway tidak memiliki izin yang diperlukan untuk menjalankan fungsi Lambda.

Saat Anda membuat integrasi Lambda di konsol API Gateway, API Gateway secara otomatis mengonfigurasi izin untuk menjalankan fungsi Lambda. Saat membuat integrasi Lambda dengan menggunakan AWS CLI, atau SDK CloudFormation, Anda harus memberikan izin kepada API Gateway untuk menjalankan fungsi tersebut. Perintah [izin tambahan](#) berikut memberikan izin untuk rute API HTTP yang berbeda untuk menjalankan fungsi Lambda.

Example Contoh - Untuk **\$default** tahap dan **\$default** rute API HTTP

```
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 Contoh - Untuk **prod** tahap dan **test** rute API HTTP

```
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"
```

[Konfirmasi kebijakan fungsi](#) di tab Izin di konsol Lambda.

Coba jalankan API Anda lagi. Anda akan melihat respons fungsi Lambda Anda.

Memecahkan masalah dengan otorisasi HTTP API JWT

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan otorisasi JSON Web Token (JWT) dengan HTTP. APIs

Masalah: API saya kembali **401 {"message":"Unauthorized"}**

Periksa `www-authenticate` header dalam respons dari API.

Perintah berikut digunakan `curl` untuk mengirim permintaan ke API dengan otorisasi JWT yang digunakan `$request.header.Authorization` sebagai sumber identitasnya.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-west-2.amazonaws.com/route
```

Respons dari API termasuk `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"}}
```

Dalam hal ini, `www-authenticate` header menunjukkan bahwa token tidak dikeluarkan untuk audiens yang valid. Agar API Gateway dapat mengotorisasi permintaan, JWT `aud` atau `client_id` klaim harus cocok dengan salah satu entri audiens yang dikonfigurasi untuk otorisasi. API Gateway `client_id` hanya memvalidasi jika tidak `aud` ada. Saat keduanya `aud` dan `client_id` ada, API Gateway mengevaluasi `aud`.

Anda juga dapat memecahkan kode JWT dan memverifikasi bahwa JWT cocok dengan penerbit, audiens, dan cakupan yang dibutuhkan API Anda. Situs web jwt.io dapat melakukan debug JWTs di browser. OpenID Foundation menyimpan [daftar pustaka untuk dikerjakan](#). JWTs

Untuk mempelajari selengkapnya tentang otorisasi JWT, lihat. [Kontrol akses ke HTTP APIs dengan otorisasi JWT di API Gateway](#)

API Gateway WebSocket APIs

WebSocket API di API Gateway adalah kumpulan WebSocket rute yang terintegrasi dengan titik akhir HTTP backend, fungsi Lambda, atau layanan lainnya. AWS Anda dapat menggunakan fitur API Gateway untuk membantu Anda dengan semua aspek siklus hidup API, mulai dari pembuatan hingga memantau produksi Anda. APIs

API Gateway WebSocket APIs bersifat dua arah. Klien dapat mengirim pesan ke layanan, dan layanan dapat secara mandiri mengirim pesan ke klien. Perilaku dua arah ini memungkinkan interaksi klien/layanan yang lebih kaya karena layanan dapat mendorong data ke klien tanpa mengharuskan klien untuk membuat permintaan eksplisit. WebSocket APIs sering digunakan dalam aplikasi real-time seperti aplikasi obrolan, platform kolaborasi, game multipemain, dan platform perdagangan keuangan.

Untuk contoh aplikasi untuk memulai, lihat [Tutorial: Buat aplikasi WebSocket obrolan dengan WebSocket API, Lambda, dan DynamoDB](#).

Di bagian ini, Anda dapat mempelajari cara mengembangkan, menerbitkan, melindungi, dan memantau WebSocket APIs penggunaan API Gateway.

Topik

- [Ikhtisar WebSocket APIs di API Gateway](#)
- [Kembangkan WebSocket APIs di API Gateway](#)
- [Publikasikan WebSocket APIs untuk pelanggan untuk dipanggil](#)
- [Lindungi Anda WebSocket APIs di API Gateway](#)
- [Pantau WebSocket APIs di API Gateway](#)

Ikhtisar WebSocket APIs di API Gateway

Di API Gateway Anda dapat membuat WebSocket API sebagai frontend stateful untuk AWS layanan (seperti Lambda atau DynamoDB) atau untuk titik akhir HTTP. WebSocket API memanggil backend Anda berdasarkan konten pesan yang diterimanya dari aplikasi klien.

Tidak seperti REST API, yang menerima dan merespons permintaan, WebSocket API mendukung komunikasi dua arah antara aplikasi klien dan backend Anda. Backend dapat mengirim pesan callback ke klien yang terhubung.

Di WebSocket API Anda, pesan JSON yang masuk diarahkan ke integrasi backend berdasarkan rute yang Anda konfigurasi. (Pesan non-JSON diarahkan ke `$default` rute yang Anda konfigurasi.)

Rute menyertakan kunci rute, yang merupakan nilai yang diharapkan setelah ekspresi pemilihan rute dievaluasi. `routeSelectionExpression` adalah atribut yang didefinisikan pada tingkat API. Ini menentukan properti JSON yang diharapkan hadir dalam payload pesan. Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihat [the section called “”](#).

Misalnya, jika pesan JSON Anda berisi `action` properti, dan Anda ingin melakukan tindakan berbeda berdasarkan properti ini, ekspresi pemilihan rute Anda mungkin `{request.body.action}`. Tabel routing Anda akan menentukan tindakan mana yang akan dilakukan dengan mencocokkan nilai `action` properti terhadap nilai kunci rute kustom yang telah Anda tentukan dalam tabel.

Menggunakan rute untuk WebSocket API

Ada tiga rute standar yang dapat digunakan: `$connect`, `$disconnect`, dan `$default`. Selain itu, Anda dapat membuat rute khusus.

- API Gateway memanggil `$connect` rute saat koneksi persisten antara klien dan WebSocket API sedang dimulai.
- API Gateway memanggil `$disconnect` rute saat klien atau server terputus dari API.
- API Gateway memanggil rute kustom setelah ekspresi pemilihan rute dievaluasi terhadap pesan jika rute yang cocok ditemukan; kecocokan menentukan integrasi mana yang dipanggil.
- API Gateway memanggil `$default` rute jika ekspresi pemilihan rute tidak dapat dievaluasi terhadap pesan atau jika tidak ditemukan rute yang cocok.

Untuk informasi selengkapnya tentang `$disconnect` rute `$connect` dan rute, lihat [the section called “Mengelola pengguna yang terhubung dan aplikasi klien”](#).

Untuk informasi selengkapnya tentang `$default` rute dan rute khusus, lihat [the section called “Panggil integrasi backend Anda”](#).

Mengirim data ke aplikasi klien yang terhubung

Layanan backend dapat mengirim data ke aplikasi klien yang terhubung. Anda dapat mengirim data dengan melakukan hal berikut:

- Gunakan integrasi untuk mengirim respons, yang dikembalikan ke klien dengan respons rute yang telah Anda tentukan.
- Anda dapat menggunakan @connections API untuk mengirim permintaan POST. Untuk informasi selengkapnya, lihat [the section called “Gunakan @connections perintah di layanan backend Anda”](#).

WebSocket Kode status API

API Gateway WebSocket APIs menggunakan kode status berikut untuk komunikasi dari server ke klien seperti yang dijelaskan dalam [WebSocket Tutup Registri Nomor Kode](#):

1001

API Gateway mengembalikan kode status ini saat klien menganggur selama 10 menit atau mencapai masa pakai koneksi maksimum 2 jam.

1003

API Gateway mengembalikan kode status ini ketika titik akhir menerima tipe media biner. Jenis media biner tidak didukung untuk WebSocket APIs.

1005

API Gateway mengembalikan kode status ini jika klien mengirimkan bingkai tutup tanpa kode penutupan.

1006

API Gateway mengembalikan kode status ini jika ada penutupan koneksi yang tidak terduga, seperti koneksi TCP ditutup tanpa bingkai WebSocket tutup.

1008

API Gateway mengembalikan kode status ini ketika titik akhir menerima terlalu banyak permintaan dari klien tertentu.

1009

API Gateway mengembalikan kode status ini ketika titik akhir menerima pesan yang terlalu besar untuk diproses.

1011

API Gateway mengembalikan kode status ini ketika ada kesalahan server internal.

1012

API Gateway mengembalikan kode status ini jika layanan dimulai ulang.

Mengelola pengguna yang terhubung dan aplikasi klien: **\$connect** dan **\$disconnect** rute

Bagian berikut menjelaskan cara menggunakan `$connect` dan `$disconnect` rute untuk WebSocket API Anda.

Topik

- [\\$connectRoute](#)
- [Melewati informasi koneksi dari \\$connect rute](#)
- [\\$disconnectRoute](#)

\$connectRoute

Aplikasi klien terhubung ke WebSocket API Anda dengan mengirimkan permintaan WebSocket pemutakhiran. Jika permintaan berhasil, `$connect` rute dijalankan saat koneksi sedang dibuat.

Karena WebSocket koneksi adalah koneksi stateful, Anda dapat mengonfigurasi otorisasi pada rute saja. `$connect` AuthN/AuthZ akan dilakukan hanya pada waktu koneksi.

Sampai eksekusi integrasi yang terkait dengan `$connect` rute selesai, permintaan pemutakhiran tertunda dan koneksi sebenarnya tidak akan dibuat. Jika `$connect` permintaan gagal (misalnya, karena AuthN AuthZ /kegagalan atau kegagalan integrasi), koneksi tidak akan dibuat.

Note

Jika otorisasi gagal `$connect`, koneksi tidak akan dibuat, dan klien akan menerima 401 atau 403 tanggapan.

Menyiapkan integrasi untuk `$connect` adalah opsional. Anda harus mempertimbangkan untuk menyiapkan `$connect` integrasi jika:

- Anda ingin mengaktifkan klien untuk menentukan subprotokol dengan menggunakan bidang. Sec-WebSocket-Protocol Untuk kode sampel, lihat [Siapkan \\$connect rute yang membutuhkan WebSocket subprotokol](#).
- Anda ingin diberi tahu saat klien terhubung.
- Anda ingin membatasi koneksi atau mengontrol siapa yang terhubung.
- Anda ingin backend Anda mengirim pesan kembali ke klien menggunakan URL callback.
- Anda ingin menyimpan setiap ID koneksi dan informasi lainnya ke dalam database (misalnya, Amazon DynamoDB).

Melewati informasi koneksi dari **\$connect** rute

Anda dapat menggunakan integrasi proxy dan non-proxy untuk meneruskan informasi dari \$connect rute ke database atau lainnya. Layanan AWS

Untuk meneruskan informasi koneksi menggunakan integrasi proxy

Anda dapat mengakses informasi koneksi dari integrasi proxy Lambda dalam acara tersebut. Gunakan AWS Lambda fungsi lain Layanan AWS atau untuk memposting ke koneksi.

Fungsi Lambda berikut menunjukkan cara menggunakan requestContext objek untuk mencatat ID koneksi, nama domain, nama panggung, dan string kueri.

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}
```

Untuk meneruskan informasi koneksi menggunakan integrasi non-proxy

- Anda dapat mengakses informasi koneksi dengan integrasi non-proxy. Siapkan permintaan integrasi dan berikan template permintaan WebSocket API. Template pemetaan [Velocity Template Language \(VTL\)](#) berikut menyediakan permintaan integrasi. Permintaan ini mengirimkan rincian berikut ke integrasi non-proxy:
 - ID Koneksi
 - Nama domain
 - Nama panggung
 - Jalan
 - Header
 - String pertanyaan

Permintaan ini mengirimkan ID koneksi, nama domain, nama panggung, jalur, header, dan string kueri ke integrasi non-proxy.

```
{
  "connectionId": "$context.connectionId",
  "domain": "$context.domainName",
  "stage": "$context.stage",
  "params": "$input.params()"
}
```

Untuk informasi selengkapnya tentang pengaturan transformasi data, lihat [the section called “Transformasi data”](#).

Untuk menyelesaikan permintaan integrasi, `Status Code: 200` tetapkan respons integrasi.

Untuk mempelajari lebih lanjut tentang menyiapkan respons integrasi, lihat [Menyiapkan respons integrasi menggunakan konsol API Gateway](#).

`$disconnect` Route

`$disconnect` Route dijalankan setelah koneksi ditutup.

Koneksi dapat ditutup oleh server atau oleh klien. Karena koneksi sudah ditutup saat dijalankan, `$disconnect` adalah cara upaya terbaik. API Gateway akan mencoba yang terbaik untuk mengirimkan `$disconnect` acara ke integrasi Anda, tetapi tidak dapat menjamin pengiriman.

Backend dapat memulai pemutusan dengan menggunakan API. `@connections` Untuk informasi selengkapnya, lihat [the section called “Gunakan @connections perintah di layanan backend Anda”](#).

Memanggil integrasi backend Anda dengan **`$default`** Route dan rute khusus di API Gateway

Bagian berikut menjelaskan cara memanggil integrasi backend Anda menggunakan `$default` route atau rute khusus untuk API. WebSocket

Topik

- [Menggunakan rute untuk memproses pesan](#)
- [\\$defaultRoute](#)
- [Rute khusus](#)
- [Menggunakan integrasi API Gateway WebSocket API untuk terhubung ke logika bisnis Anda](#)
- [Perbedaan penting antara WebSocket APIs dan REST APIs](#)

Menggunakan rute untuk memproses pesan

Di API Gateway WebSocket APIs, pesan dapat dikirim dari klien ke layanan backend Anda dan sebaliknya. Tidak seperti request/response model HTTP, WebSocket di backend dapat mengirim pesan ke klien tanpa klien mengambil tindakan apa pun.

Pesan dapat berupa JSON atau non-JSON. Namun, hanya pesan JSON yang dapat dialihkan ke integrasi tertentu berdasarkan konten pesan. Pesan non-JSON diteruskan ke backend melalui rute. `$default`

 Note

API Gateway mendukung muatan pesan hingga 128 KB dengan ukuran bingkai maksimum 32 KB. Jika pesan melebihi 32 KB, Anda harus membaginya menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Jika pesan yang lebih besar (atau bingkai) diterima, koneksi ditutup dengan kode 1009.

Saat ini payload biner tidak didukung. Jika bingkai biner diterima, koneksi ditutup dengan kode 1003. Namun, dimungkinkan untuk mengonversi muatan biner menjadi teks. Lihat [the section called “Jenis media biner”](#).

WebSocket APIs Dengan API Gateway, pesan JSON dapat dirutekan untuk menjalankan layanan backend tertentu berdasarkan konten pesan. Ketika klien mengirim pesan melalui WebSocket koneksinya, ini menghasilkan permintaan rute ke WebSocket API. Permintaan akan dicocokkan dengan rute dengan kunci rute yang sesuai di API Gateway. Anda dapat menyiapkan permintaan rute untuk WebSocket API di konsol API Gateway, dengan menggunakan AWS CLI, atau dengan menggunakan AWS SDK.

 Note

Di AWS CLI dan AWS SDKs, Anda dapat membuat rute sebelum atau setelah Anda membuat integrasi. Saat ini konsol tidak mendukung penggunaan kembali integrasi, jadi Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi untuk rute itu.

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi pada permintaan rute sebelum melanjutkan dengan permintaan integrasi. Jika validasi gagal, API Gateway gagal permintaan tanpa memanggil backend Anda, mengirimkan respons "Bad request body" gateway yang serupa dengan yang berikut ke klien, dan menerbitkan hasil validasi di Log: CloudWatch

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId": "{messageId}"}
```

Ini mengurangi panggilan yang tidak perlu ke backend Anda dan memungkinkan Anda fokus pada persyaratan lain dari API Anda.

Anda juga dapat menentukan respons rute untuk rute API Anda untuk mengaktifkan komunikasi dua arah. Respons rute menjelaskan data apa yang akan dikirim ke klien Anda setelah menyelesaikan integrasi rute tertentu. Tidak perlu menentukan respons untuk rute jika, misalnya, Anda ingin klien mengirim pesan ke backend Anda tanpa menerima respons (komunikasi satu arah). Namun, jika Anda tidak memberikan respons rute, API Gateway tidak akan mengirimkan informasi apa pun tentang hasil integrasi Anda ke klien Anda.

\$defaultRoute

Setiap API Gateway WebSocket API dapat memiliki `$default` rute. Ini adalah nilai routing khusus yang dapat digunakan dengan cara-cara berikut:

- Anda dapat menggunakannya bersama dengan kunci rute yang ditentukan, untuk menentukan rute “fallback” (misalnya, integrasi tiruan generik yang mengembalikan pesan kesalahan tertentu) untuk pesan masuk yang tidak cocok dengan kunci rute yang ditentukan.
- Anda dapat menggunakannya tanpa kunci rute yang ditentukan, untuk menentukan model proxy yang mendelegasikan perutean ke komponen backend.
- Anda dapat menggunakannya untuk menentukan rute untuk muatan non-JSON.

Rute khusus

Jika Anda ingin menjalankan integrasi tertentu berdasarkan konten pesan, Anda dapat melakukannya dengan membuat rute khusus.

Rute kustom menggunakan kunci rute dan integrasi yang Anda tentukan. Saat pesan masuk berisi properti JSON, dan properti tersebut mengevaluasi nilai yang cocok dengan nilai kunci rute, API Gateway akan memanggil integrasi. (Untuk informasi selengkapnya, lihat [the section called “Ikhtisar WebSocket APIs”](#).)

Misalnya, Anda ingin membuat aplikasi ruang obrolan. Anda dapat memulai dengan membuat WebSocket API yang ekspresi pemilihan rutanya `$request.body.action`. Anda kemudian dapat menentukan dua rute: `joinroom` dan `sendmessage`. Aplikasi klien mungkin memanggil `joinroom` rute dengan mengirim pesan seperti berikut:

```
{"action":"joinroom","roomname":"developers"}
```


Dan itu mungkin memanggil `sendmessage` rute dengan mengirim pesan seperti berikut:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Menggunakan integrasi API Gateway WebSocket API untuk terhubung ke logika bisnis Anda

Setelah menyiapkan rute untuk API Gateway WebSocket API, Anda harus menentukan integrasi yang ingin Anda gunakan. Seperti halnya rute, yang dapat memiliki permintaan rute dan respons rute, integrasi dapat memiliki permintaan integrasi dan respons integrasi. Permintaan integrasi berisi informasi yang diharapkan oleh backend Anda untuk memproses permintaan yang berasal dari klien Anda. Respons integrasi berisi data yang dikembalikan backend Anda ke API Gateway, dan yang dapat digunakan untuk membuat pesan yang akan dikirim ke klien (jika respons rute ditentukan).

Untuk informasi selengkapnya tentang menyiapkan integrasi, lihat [the section called “Integrasi”](#).

Perbedaan penting antara WebSocket APIs dan REST APIs

Integrasi untuk WebSocket APIs mirip dengan integrasi untuk REST APIs, kecuali untuk perbedaan berikut:

- Saat ini, di konsol API Gateway Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi sebagai target rute tersebut. Namun, di API dan CLI, Anda dapat membuat rute dan integrasi secara independen, dalam urutan apa pun.
- Anda dapat menggunakan integrasi tunggal untuk beberapa rute. Misalnya, jika Anda memiliki serangkaian tindakan yang terkait erat satu sama lain, Anda mungkin ingin semua rute tersebut menuju ke satu fungsi Lambda. Daripada mendefinisikan detail integrasi beberapa kali, Anda dapat menentukannya sekali dan menetapkan ke setiap rute terkait.

Note

Saat ini konsol tidak mendukung penggunaan kembali integrasi, jadi Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi untuk rute itu.

Di AWS CLI dan AWS SDKs, Anda dapat menggunakan kembali integrasi dengan menyetel target rute ke nilai `"integrations/{integration-id}"`, di `{integration-id}` mana ID unik integrasi yang akan dikaitkan dengan rute.

- API Gateway menyediakan beberapa [ekspresi pilihan](#) yang dapat Anda gunakan dalam rute dan integrasi Anda. Anda tidak perlu bergantung pada jenis konten untuk memilih template input atau

pemetaan output. Seperti ekspresi pemilihan rute, Anda dapat menentukan ekspresi seleksi yang akan dievaluasi oleh API Gateway untuk memilih item yang tepat. Semuanya akan kembali ke `$default` template jika template yang cocok tidak ditemukan.

- Dalam permintaan integrasi, ekspresi pemilihan template mendukung `$request.body.<json_path_expression>` dan nilai statis.
- Dalam tanggapan integrasi, ekspresi pemilihan template mendukung `$request.body.<json_path_expression>`, `$integration.response.statuscode`, `$integration.response.header.<headerName>`, dan nilai statis.

Dalam protokol HTTP, di mana permintaan dan tanggapan dikirim secara serempak; komunikasi pada dasarnya satu arah. Dalam WebSocket protokol, komunikasi adalah dua arah. Respons bersifat asinkron dan tidak harus diterima oleh klien dalam urutan yang sama dengan pesan klien yang dikirim. Selain itu, backend dapat mengirim pesan ke klien.

Note

Untuk rute yang dikonfigurasi untuk digunakan `AWS_PROXY` atau `LAMBDA_PROXY` diintegrasikan, komunikasi dilakukan satu arah, dan API Gateway tidak akan meneruskan respons backend ke respons rute secara otomatis. Misalnya, dalam kasus `LAMBDA_PROXY` integrasi, badan yang dikembalikan fungsi Lambda tidak akan dikembalikan ke klien. Jika Anda ingin klien menerima tanggapan integrasi, Anda harus menentukan respons rute untuk memungkinkan komunikasi dua arah.

WebSocket ekspresi seleksi

API Gateway menggunakan ekspresi seleksi sebagai cara untuk mengevaluasi konteks permintaan dan respons dan menghasilkan kunci. Kunci tersebut kemudian digunakan untuk memilih dari serangkaian nilai yang mungkin, biasanya disediakan oleh Anda, pengembang API. Kumpulan yang tepat dari variabel yang didukung akan bervariasi tergantung pada ekspresi tertentu. Setiap ekspresi dibahas secara lebih rinci di bawah ini.

Untuk semua ekspresi, bahasa mengikuti seperangkat aturan yang sama:

- Variabel diawali dengan "\$".

- Kurung kurawal dapat digunakan untuk secara eksplisit mendefinisikan batas variabel, misalnya, `"${request.body.version}-beta"`
- Beberapa variabel didukung, tetapi evaluasi hanya terjadi sekali (tidak ada evaluasi rekursif).
- Tanda dolar (\$) dapat lolos dengan `"\"`. Ini paling berguna saat mendefinisikan ekspresi yang memetakan ke `$default` kunci yang dicadangkan, misalnya, `"\"$default"`.
- Dalam beberapa kasus, format pola diperlukan. Dalam hal ini, ekspresi harus dibungkus dengan garis miring (`"/"`), misalnya `"/2\d\d/"` untuk mencocokkan kode `2XX` status.

Topik

- [Ekspresi pemilihan respons rute](#)
- [Ekspresi pemilihan kunci API](#)
- [Ekspresi pemilihan pemetaan API](#)
- [WebSocket ringkasan ekspresi seleksi](#)

Ekspresi pemilihan respons rute

[Respons rute](#) digunakan untuk memodelkan respons dari backend ke klien. Untuk WebSocket APIs, respons rute adalah opsional. Ketika didefinisikan, ini memberi sinyal ke API Gateway bahwa ia harus mengembalikan respons ke klien setelah menerima WebSocket pesan.

Evaluasi ekspresi pemilihan respons rute menghasilkan kunci respons rute. Akhirnya, kunci ini akan digunakan untuk memilih dari salah satu yang [RouteResponses](#) terkait dengan API. Namun, saat ini hanya `$default` kunci yang didukung.

Ekspresi pemilihan kunci API

Ekspresi ini dievaluasi ketika layanan menentukan permintaan yang diberikan harus dilanjutkan hanya jika klien menyediakan [kunci API](#) yang valid.

Saat ini hanya dua nilai yang didukung adalah `$request.header.x-api-key` dan `$context.authorizer.usageIdentifierKey`.

Ekspresi pemilihan pemetaan API

Ekspresi ini dievaluasi untuk menentukan tahap API mana yang dipilih saat permintaan dibuat menggunakan domain kustom.

Saat ini, satu-satunya nilai yang didukung adalah `$request.basepath`.

WebSocket ringkasan ekspresi seleksi

Tabel berikut merangkum kasus penggunaan untuk ekspresi seleksi di WebSocket APIs:

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
<code>Api.RouteSelectionExpression</code>	<code>Route.RouteKey</code>	<code>\$default</code> sebagai rute catch-all.	Rute yang digunakan sebagai WebSocket pesan berdasarkan konteks permintaan klien.
<code>Route.ModelSelectionExpression</code>	Kunci untuk <code>Route.RequestModels</code>	Tidak wajib. Jika disediakan untuk integrasi non-proxy, validasi model terjadi. <code>\$default</code> sebagai catch-all.	Lakukan validasi permintaan secara dinamis dalam rute yang sama.

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
Integration.TemplateSelectionExpression	Kunci untuk <code>Integration.RequestTemplates</code>	Tidak wajib. Dapat disediakan untuk integrasi non-proxy untuk memanipulasi muatan yang masuk. <code>\${request.body.jsonPath}</code> nilai statis didukung <code>\$default</code> sebagai catch-all.	Memanipulasi permintaan pemanggilan berdasarkan properti dinamis permintaan. g

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
IntegrationResponse.IntegrationResponseSelectionExpression	IntegrationResponse.IntegrationResponseKey	Tidak wajib. Dapat disediakan untuk integrasi non-proxy. Bertindak sebagai kecocokan pola untuk pesan kesalahan (dari Lambda) atau kode status (dari integrasi HTTP). \$default untuk integrasi non-proxy	Memanipulasi respons dari backend. Pilih tindakan yang akan terjadi berdasarkan an respons dinamis backend (misalnya , menangani kesalahan tertentu dengan jelas). a

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
		untuk bertindak sebagai tangkapan semua untuk tangkapan yang berhasil.	

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
IntegrationResponse.TemplateSelectionExpression	Kunci untuk IntegrationResponse.ResponseTemplates	Tidak wajib. Dapat disediakan untuk integrasi non-proxy. \$ default didukung	Dalam beberapa kasus, properti dinamis dari respons dapat menentukan transformasi yang berbeda dalam rute yang sama dan integrasi terkait. <code>\${request.body.jsonPath} ,\${integration.response.statuscode} ,\${integration.response.header.headerName} ,\${int</code>

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
			<code>tion.response.multiValueHeader.headerName} ,</code> dan nilai statis didukung. <code>\$defaultdidul</code> sebagai catch- all.

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaa n
<code>Route.RouteResponseSelectionExpression</code>	<code>RouteResponse.RouteResponseKey</code>	Harus disediakan untuk memulai komunikasi dua arah untuk suatu WebSocket route. Saat ini, nilai ini dibatasi <code>\$default</code> hanya untuk.	
<code>RouteResponse.ModelSelectionExpression</code>	Kunci untuk <code>RouteResponse.RequestModels</code>	Saat ini tidak didukung	

Kembangkan WebSocket APIs di API Gateway

Bagian ini memberikan detail tentang kemampuan API Gateway yang Anda butuhkan saat mengembangkan API Gateway APIs.

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Misalnya, Anda mungkin hanya ingin mengizinkan klien tertentu untuk memanggil API Anda, atau Anda mungkin ingin itu tersedia untuk semua orang. Anda mungkin ingin panggilan API untuk menjalankan fungsi Lambda, membuat kueri database, atau memanggil aplikasi.

Topik

- [Buat WebSocket APIs di API Gateway](#)
- [Jenis alamat IP untuk WebSocket APIs di API Gateway](#)
- [Buat rute untuk WebSocket APIs di API Gateway](#)
- [Mengontrol dan mengelola akses ke WebSocket APIs dalam API Gateway](#)
- [Integrasi untuk WebSocket APIs di API Gateway](#)
- [Minta validasi untuk WebSocket APIs di API Gateway](#)
- [Transformasi data untuk WebSocket APIs di API Gateway](#)
- [Jenis media biner untuk WebSocket APIs di API Gateway](#)
- [Memohon WebSocket APIs](#)

Buat WebSocket APIs di API Gateway

Anda dapat membuat WebSocket API di konsol API Gateway, dengan menggunakan perintah AWS CLI [create-api](#), atau dengan menggunakan `CreateApi` perintah dalam SDK. AWS Prosedur berikut menunjukkan cara membuat WebSocket API baru.

Note

WebSocket APIs hanya mendukung TLS 1.2 dan TLS 1.3. Versi TLS sebelumnya tidak didukung.

Buat WebSocket API menggunakan AWS CLI perintah

Perintah [create-api](#) berikut membuat API dengan ekspresi pemilihan `$request.body.action` rute:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
WEBSOCKET --route-selection-expression '$request.body.action'
```

Outputnya terlihat seperti berikut:

```
{
  "ApiKeySelectionExpression": "$request.header.x-api-key",
  "Name": "myWebSocketApi3",
  "CreateDate": "2018-11-15T06:23:51Z",
  "ProtocolType": "WEBSOCKET",
  "RouteSelectionExpression": "'$request.body.action'",
  "ApiId": "aabbccdde"
}
```

Membuat WebSocket API menggunakan konsol API Gateway

Anda dapat membuat WebSocket API di konsol dengan memilih WebSocket protokol dan memberi API nama.

Important

Setelah Anda membuat API, Anda tidak dapat mengubah protokol yang telah Anda pilih untuk itu. Tidak ada cara untuk mengubah WebSocket API menjadi REST API atau sebaliknya.

Untuk membuat WebSocket API menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway dan pilih Buat API.
2. Di bawah WebSocket API, pilih Build. Hanya titik akhir Regional yang didukung.
3. Untuk nama API, masukkan nama API Anda.
4. Untuk ekspresi pemilihan Rute, masukkan nilai. Misalnya, `$request.body.action`.

Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihat [the section called “”](#).

5. Lakukan salah satu tindakan berikut:
 - Pilih Buat API kosong untuk membuat API tanpa rute.
 - Pilih Berikutnya untuk melampirkan rute ke API Anda.

Anda dapat melampirkan rute setelah membuat API.

Jenis alamat IP untuk WebSocket APIs di API Gateway

Saat membuat API, Anda menentukan jenis alamat IP yang dapat menjalankan API Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk menjalankan API Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat menjalankan API Anda. Kami menyarankan Anda mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda:

- Jenis alamat IP default untuk semua WebSocket APIs adalah IPv4.
- Jika Anda mengubah jenis alamat IP untuk API yang ada dari IPv4 ke dualstack, konfirmasi bahwa kebijakan apa pun yang mengontrol akses ke Anda APIs telah diperbarui untuk IPv6 memperhitungkan panggilan. Ketika Anda mengubah jenis alamat IP, perubahan akan segera berlaku.
- API Anda dapat dipetakan ke nama domain khusus dengan jenis alamat IP yang berbeda dari API Anda. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.

Mengubah jenis alamat IP WebSocket API

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi API. Anda dapat memperbarui konfigurasi API dengan menggunakan Konsol Manajemen AWS, the AWS CLI CloudFormation, atau AWS SDK. Jika Anda mengubah jenis alamat IP API, Anda tidak menerapkan ulang API agar perubahan diterapkan.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP WebSocket API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih WebSocket API.
3. Pilih pengaturan API, lalu pilih Edit.
4. Untuk jenis alamat IP, pilih salah satu IPv4atau Dualstack.

5. Pilih Simpan.

Perubahan pada konfigurasi API Anda akan segera berlaku.

AWS CLI

Perintah [update-api berikut memperbarui API](#) agar memiliki tipe alamat IP dualstack:

```
aws apigatewayv2 update-api \  
  --api-id abcd1234 \  
  --ip-address-type dualstack
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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"  
}
```

Buat rute untuk WebSocket APIs di API Gateway

Di WebSocket API Anda, pesan JSON yang masuk diarahkan ke integrasi backend berdasarkan rute yang Anda konfigurasi. (Pesan non-JSON diarahkan ke `$default` rute yang Anda konfigurasi.)

Rute menyertakan kunci rute, yang merupakan nilai yang diharapkan setelah ekspresi pemilihan rute dievaluasi. `routeSelectionExpression` ini adalah atribut yang didefinisikan pada tingkat API. Ini menentukan properti JSON yang diharapkan hadir dalam payload pesan. Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihat [the section called “”](#).

Misalnya, jika pesan JSON Anda berisi `action` properti dan Anda ingin melakukan tindakan berbeda berdasarkan properti ini, ekspresi pemilihan rute Anda mungkin `${request.body.action}`. Tabel routing Anda akan menentukan tindakan mana yang akan dilakukan dengan mencocokkan nilai `action` properti terhadap nilai kunci rute kustom yang telah Anda tentukan dalam tabel.

Ada tiga rute standar yang dapat digunakan: `$connect`, `$disconnect`, dan `$default`. Selain itu, Anda dapat membuat rute khusus.

- API Gateway memanggil `$connect` rute saat koneksi persisten antara klien dan WebSocket API sedang dimulai.
- API Gateway memanggil `$disconnect` rute saat klien atau server terputus dari API.
- API Gateway memanggil rute kustom setelah ekspresi pemilihan rute dievaluasi terhadap pesan jika rute yang cocok ditemukan; kecocokan menentukan integrasi mana yang dipanggil.
- API Gateway memanggil `$default` rute jika ekspresi pemilihan rute tidak dapat dievaluasi terhadap pesan atau jika tidak ditemukan rute yang cocok.

Ekspresi pemilihan rute

Ekspresi pemilihan rute dievaluasi saat layanan memilih rute yang akan diikuti untuk pesan masuk. Layanan menggunakan rute yang `routeKey` persis sama dengan nilai yang dievaluasi. Jika tidak ada yang cocok dan rute dengan kunci `$default` rute ada, rute itu dipilih. Jika tidak ada rute yang cocok dengan nilai yang dievaluasi dan tidak ada `$default` rute, layanan mengembalikan kesalahan. Untuk WebSocket berbasis APIs, ekspresi harus dari bentuk `${request.body.{path_to_body_element}}`.

Misalnya, Anda mengirim pesan JSON berikut:

```
{
  "service" : "chat",
  "action" : "join",
  "data" : {
    "room" : "room1234"
  }
}
```

Anda mungkin ingin memilih perilaku API berdasarkan `action` properti. Dalam hal ini, Anda dapat menentukan ekspresi pemilihan rute berikut:

```
$request.body.action
```

Dalam contoh ini, `request.body` mengacu pada payload JSON pesan Anda, dan `.action` merupakan [JSONPath](#) ekspresi. Anda dapat menggunakan ekspresi jalur JSON apa pun setelahnya `request.body`, tetapi perlu diingat bahwa hasilnya akan dirangkai. Misalnya, jika JSONPath ekspresi Anda mengembalikan array dari dua elemen, yang akan disajikan sebagai `string"[item1, item2]"`. Untuk alasan ini, ini adalah praktik yang baik untuk membuat ekspresi Anda mengevaluasi nilai dan bukan array atau objek.

Anda cukup menggunakan nilai statis, atau Anda dapat menggunakan beberapa variabel. Tabel berikut menunjukkan contoh dan hasil evaluasi mereka terhadap muatan sebelumnya.

Ekspresi	Hasil yang dievaluasi	Deskripsi
<code>\$request.body.action</code>	<code>join</code>	Variabel yang tidak dibungkus
<code>\${request.body.action}</code>	<code>join</code>	Variabel yang dibungkus
<code>\${request.body.service}/\${request.body.action}</code>	<code>chat/join</code>	Beberapa variabel dengan nilai statis
<code>\${request.body.action}-\${request.body.invalidPath}</code>	<code>join-</code>	Jika JSONPath tidak ditemukan, variabel diselesaikan

Ekspresi	Hasil yang dievaluasi	Deskripsi
		sebagai “” .
action	action	Nilai statis
\\$default	\$default	Nilai statis

Hasil evaluasi digunakan untuk menemukan rute. Jika ada rute dengan kunci rute yang cocok, rute dipilih untuk memproses pesan. Jika tidak ditemukan rute yang cocok, API Gateway mencoba menemukan \$default rute jika tersedia. Jika \$default rute tidak ditentukan, maka API Gateway mengembalikan kesalahan.

Mengatur rute untuk WebSocket API di API Gateway

Saat pertama kali membuat WebSocket API baru, ada tiga rute yang telah ditentukan: \$connect, \$disconnect, dan \$default. Anda dapat membuatnya dengan menggunakan konsol, API, atau AWS CLI. Jika diinginkan, Anda dapat membuat rute khusus. Untuk informasi selengkapnya, lihat [the section called “Ikhtisar WebSocket APIs”](#).

Note

Di CLI, Anda dapat membuat rute sebelum atau setelah Anda membuat integrasi, dan Anda dapat menggunakan kembali integrasi yang sama untuk beberapa rute.

Membuat rute menggunakan konsol API Gateway

Untuk membuat rute menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Pilih Buat rute
3. Untuk tombol Rute, masukkan nama kunci rute. Anda dapat membuat rute yang telah ditentukan (\$connect, \$disconnect, dan \$default), atau rute khusus.

 Note

Saat Anda membuat rute khusus, jangan gunakan \$ awalan dalam nama kunci rute. Awalan ini dicadangkan untuk rute yang telah ditentukan sebelumnya.

4. Pilih dan konfigurasi jenis integrasi untuk rute. Untuk informasi selengkapnya, lihat [the section called “Menyiapkan permintaan integrasi WebSocket API menggunakan konsol API Gateway”](#).

Buat rute menggunakan AWS CLI

Perintah [create-route](#) berikut membuat rute:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccdde --route-key $default
```

Outputnya akan terlihat seperti berikut:

```
{
  "ApiKeyRequired": false,
  "AuthorizationType": "NONE",
  "RouteKey": "$default",
  "RouteId": "1122334"
}
```

Tentukan pengaturan permintaan rute untuk **\$connect**

Saat Anda menyiapkan \$connect rute untuk API Anda, pengaturan opsional berikut tersedia untuk mengaktifkan otorisasi API Anda. Untuk informasi selengkapnya, lihat [the section called “\\$connectRule”](#).

- Otorisasi: Jika tidak diperlukan otorisasi, Anda dapat menentukan. NONE Jika tidak, Anda dapat menentukan:
 - AWS_IAM untuk menggunakan kebijakan AWS IAM standar untuk mengontrol akses ke API Anda.
 - CUSTOM untuk menerapkan otorisasi untuk API dengan menentukan fungsi otorisasi Lambda yang telah Anda buat sebelumnya. Authorizer dapat berada di akun Anda sendiri atau AWS akun lain AWS . Untuk informasi selengkapnya tentang otorisasi Lambda, lihat. [Gunakan otorisasi API Gateway Lambda](#)

 Note

Di konsol API Gateway, CUSTOM pengaturan hanya terlihat setelah Anda menyiapkan fungsi otorisasi seperti yang dijelaskan dalam [the section called “Konfigurasi otorisasi Lambda \(konsol\)”](#).

 Important

Pengaturan Otorisasi diterapkan ke seluruh API, bukan hanya \$connect rute. \$connectRute melindungi rute lain, karena dipanggil pada setiap koneksi.

- Kunci API diperlukan: Anda dapat secara opsional meminta kunci API untuk \$connect rute API. Anda dapat menggunakan kunci API bersama dengan rencana penggunaan untuk mengontrol dan melacak akses ke Anda APIs. Untuk informasi selengkapnya, lihat [the section called “Paket penggunaan”](#).

Siapkan permintaan **\$connect** rute menggunakan konsol API Gateway

Untuk menyiapkan permintaan \$connect rute WebSocket API menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Di bawah Rute\$connect, pilih, atau buat \$connect rute dengan mengikuti [the section called “Membuat rute menggunakan konsol API Gateway”](#).
3. Di bagian Pengaturan permintaan rute, pilih Edit.
4. Untuk Otorisasi, pilih jenis otorisasi.
5. Untuk mewajibkan API untuk \$connect rute tersebut, pilih Memerlukan kunci API.
6. Pilih Simpan perubahan.

Siapkan respons rute untuk WebSocket APIs di API Gateway

WebSocket rute dapat dikonfigurasi untuk komunikasi dua arah atau satu arah. API Gateway tidak akan meneruskan respons backend ke respons rute, kecuali jika Anda menyiapkan respons rute.

 Note

Anda hanya dapat menentukan respons `$default` rute untuk WebSocket APIs. Anda dapat menggunakan respons integrasi untuk memanipulasi respons dari layanan backend. Untuk informasi selengkapnya, lihat [the section called “Ikhtisar tanggapan integrasi”](#).

Anda dapat mengonfigurasi respons rute dan ekspresi pemilihan respons menggunakan konsol API Gateway atau SDK AWS CLI atau AWS SDK.

Untuk informasi selengkapnya tentang ekspresi pemilihan respons rute, lihat [the section called “”](#).

Topik

- [Mengatur respons rute menggunakan konsol API Gateway](#)
- [Siapkan respons rute menggunakan AWS CLI](#)

Mengatur respons rute menggunakan konsol API Gateway

Setelah membuat WebSocket API dan melampirkan fungsi Lambda proxy ke rute default, Anda dapat mengatur respons rute menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway, pilih WebSocket API dengan integrasi fungsi Lambda proxy pada `$default` rute.
2. Di bawah Rute, pilih `$default` rute.
3. Pilih Aktifkan komunikasi dua arah.
4. Pilih Deploy API.
5. Menerapkan API Anda ke panggung.

Gunakan perintah [wscat](#) berikut untuk terhubung ke API Anda. Untuk informasi selengkapnya tentang `wscat`, lihat [the section called “Gunakan wscat untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Tekan tombol enter untuk memanggil rute default. Tubuh fungsi Lambda Anda harus kembali.

Siapkan respons rute menggunakan AWS CLI

[create-route-response](#) Perintah berikut membuat respon rute untuk `$default` rute. Anda dapat mengidentifikasi ID API dan ID rute dengan menggunakan perintah [get-apis](#) dan [get-routes](#).

```
aws apigatewayv2 create-route-response \  
  --api-id aabbccdde \  
  --route-id 1122334 \  
  --route-response-key '$default'
```

Outputnya akan terlihat seperti berikut:

```
{  
  "RouteResponseId": "abcdef",  
  "RouteResponseKey": "$default"  
}
```

Siapkan **\$connect** rute yang membutuhkan WebSocket subprotokol

Klien dapat menggunakan `Sec-WebSocket-Protocol` bidang untuk meminta [WebSocket subprotokol](#) selama koneksi ke WebSocket API Anda. Anda dapat menyiapkan integrasi untuk `$connect` rute untuk mengizinkan koneksi hanya jika klien meminta subprotokol yang didukung API Anda.

Contoh berikut fungsi Lambda mengembalikan `Sec-WebSocket-Protocol` header ke klien. Fungsi ini membuat koneksi ke API Anda hanya jika klien menentukan subprotokol. `myprotocol`

Untuk CloudFormation template yang membuat contoh integrasi proxy API dan Lambda ini, lihat. [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;
}

```

Anda dapat menggunakan [wscat](#) untuk menguji apakah API Anda mengizinkan koneksi hanya jika klien meminta subprotokol yang didukung API Anda. Perintah berikut menggunakan -s bendera untuk menentukan subprotokol selama koneksi.

Perintah berikut mencoba koneksi dengan subprotokol yang tidak didukung. Karena klien menentukan chat1 subprotokol, integrasi Lambda mengembalikan kesalahan 400, dan koneksi tidak berhasil.

```

wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400

```

Perintah berikut mencakup subprotokol yang didukung dalam permintaan koneksi. Integrasi Lambda memungkinkan koneksi.

```

wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)

```

Untuk mempelajari lebih lanjut tentang pemanggilan WebSocket APIs, lihat [Memohon WebSocket APIs](#).

Mengontrol dan mengelola akses ke WebSocket APIs dalam API Gateway

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke WebSocket API Anda.

Anda dapat menggunakan mekanisme berikut untuk otentikasi dan otorisasi:

- Peran dan kebijakan AWS IAM standar menawarkan kontrol akses yang fleksibel dan kuat. Anda dapat menggunakan peran dan kebijakan IAM untuk mengontrol siapa yang dapat membuat dan mengelola API, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat [Kontrol akses WebSocket APIs dengan otorisasi IAM](#).
- Tag IAM dapat digunakan bersama dengan kebijakan IAM untuk mengontrol akses. Untuk informasi selengkapnya, lihat [Menggunakan tag untuk mengontrol akses ke sumber daya API API Gateway REST API](#).
- Lambda Authorizer adalah fungsi Lambda yang mengontrol akses ke APIs. Untuk informasi selengkapnya, lihat [Kontrol akses WebSocket APIs dengan otorisasi AWS Lambda REQUEST](#).

Untuk meningkatkan postur keamanan Anda, kami sarankan Anda mengonfigurasi otorisasi untuk `$connect` rute di semua rute Anda WebSocket APIs. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Topik

- [Kontrol akses WebSocket APIs dengan otorisasi IAM](#)
- [Kontrol akses WebSocket APIs dengan otorisasi AWS Lambda REQUEST](#)

Kontrol akses WebSocket APIs dengan otorisasi IAM

Otorisasi IAM di WebSocket APIs mirip dengan yang untuk [REST APIs](#), dengan pengecualian berikut:

- `execute-api` Tindakan mendukung `ManageConnections` selain tindakan yang ada (`Invoke`, `InvalidateCache`). `ManageConnections` mengontrol akses ke `@connections` API.
- WebSocket rute menggunakan format ARN yang berbeda:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- @connectionsAPI menggunakan format ARN yang sama dengan REST: APIs

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

Important

Ketika Anda menggunakan [otorisasi IAM](#), Anda harus menandatangani permintaan dengan [Signature Version 4 \(SigV4\)](#).

Misalnya, Anda dapat menyiapkan kebijakan berikut ke klien. Contoh ini memungkinkan setiap orang untuk mengirim pesan (Invoke) untuk semua rute kecuali untuk rute rahasia di prod panggung dan mencegah semua orang mengirim pesan kembali ke klien yang terhubung (ManageConnections) untuk semua tahapan.

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/*"
      ]
    }
  ]
}

```

Kontrol akses WebSocket APIs dengan otorisasi AWS Lambda REQUEST

Fungsi otorisasi Lambda di mirip dengan WebSocket APIs yang untuk [REST APIs](#), dengan pengecualian berikut:

- Anda hanya dapat menggunakan fungsi otorisasi Lambda untuk rute tersebut. `$connect`
- Anda tidak dapat menggunakan variabel jalur (`event.pathParameters`), karena jalurnya sudah diperbaiki.
- `event.methodArn` berbeda dari setara REST API-nya, karena tidak memiliki metode HTTP. Dalam kasus `$connect`, `methodArn` diakhiri dengan `"$connect"`:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
```

- Variabel konteks `event.requestContext` berbeda dari variabel untuk REST APIs.

Contoh berikut menunjukkan input ke REQUEST authorizer untuk 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": [
      "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"
    ]
  }
}
```

```

    ]
  },
  "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"
  }
}

```

Contoh berikut fungsi Lambda authorizer adalah WebSocket versi dari fungsi Lambda authorizer untuk REST di: APIs [the section called “Contoh tambahan fungsi otorisasi Lambda”](#)

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)
```

[Untuk mengonfigurasi fungsi Lambda sebelumnya sebagai fungsi REQUEST otorisasi untuk WebSocket API, ikuti prosedur yang sama seperti REST. APIs](#)

Untuk mengonfigurasi \$connect rute untuk menggunakan otorisasi Lambda ini di konsol, pilih atau buat rute. \$connect Di bagian Pengaturan permintaan rute, pilih Edit. Pilih otorisasi Anda di menu tarik-turun Otorisasi, lalu pilih Simpan perubahan.

Untuk menguji otorisasi, Anda perlu membuat koneksi baru. Mengubah otorisasi \$connect tidak memengaruhi klien yang sudah terhubung. Saat Anda terhubung ke WebSocket API, Anda perlu memberikan nilai untuk sumber identitas yang dikonfigurasi. Misalnya, Anda dapat terhubung dengan mengirimkan string kueri dan header yang valid menggunakan wscat seperti pada contoh berikut:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Jika Anda mencoba untuk terhubung tanpa nilai identitas yang valid, Anda akan menerima 401 tanggapan:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Integrasi untuk WebSocket APIs di API Gateway

Setelah menyiapkan rute API, Anda harus mengintegrasikannya dengan titik akhir di backend. Endpoint backend juga disebut sebagai titik akhir integrasi dan dapat berupa fungsi Lambda, titik

akhir HTTP, atau tindakan layanan. AWS Integrasi API memiliki permintaan integrasi dan respons integrasi.

Di bagian ini, Anda dapat mempelajari cara menyiapkan permintaan integrasi dan respons integrasi untuk WebSocket API Anda.

Topik

- [Menyiapkan permintaan integrasi WebSocket API di API Gateway](#)
- [Menyiapkan respons integrasi WebSocket API di API Gateway](#)

Menyiapkan permintaan integrasi WebSocket API di API Gateway

Menyiapkan permintaan integrasi melibatkan hal-hal berikut:

- Memilih kunci rute untuk diintegrasikan ke backend.
- Menentukan titik akhir backend untuk dipanggil. WebSocket APIs mendukung jenis integrasi berikut:
 - AWS_PROXY
 - AWS
 - HTTP_PROXY
 - HTTP
 - MOCK

Untuk informasi selengkapnya tentang jenis integrasi, lihat [IntegrationType](#) di API REST API Gateway V2.

- Mengkonfigurasi cara mengubah data permintaan rute, jika perlu, menjadi data permintaan integrasi dengan menentukan satu atau beberapa templat permintaan.

Menyiapkan permintaan integrasi WebSocket API menggunakan konsol API Gateway

Untuk menambahkan permintaan integrasi ke rute di WebSocket API menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Di bawah Rute, pilih rute.

3. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan permintaan integrasi, pilih Edit.
4. Untuk jenis Integrasi, pilih salah satu dari berikut ini:
 - Pilih fungsi Lambda hanya jika API Anda akan diintegrasikan dengan AWS Lambda fungsi yang telah Anda buat di akun ini atau di akun lain.

Untuk membuat fungsi Lambda baru di AWS Lambda, untuk menetapkan izin sumber daya pada fungsi Lambda, atau untuk melakukan tindakan layanan Lambda lainnya, pilih Layanan sebagai gantinya.AWS

- Pilih HTTP jika API Anda akan terintegrasi dengan titik akhir HTTP yang ada. Untuk informasi selengkapnya, lihat [Integrasi HTTP untuk REST APIs di API Gateway](#).
 - Pilih Mock jika Anda ingin menghasilkan respons API dari API Gateway secara langsung, tanpa perlu backend integrasi. Untuk informasi selengkapnya, lihat [Integrasi tiruan untuk REST APIs di API Gateway](#).
 - Pilih AWS layanan jika API Anda akan terintegrasi dengan AWS layanan.
 - Pilih tautan VPC jika API Anda akan menggunakan titik akhir integrasi VpcLink sebagai pribadi. Untuk informasi selengkapnya, lihat [Siapkan integrasi pribadi](#).
5. Jika Anda memilih fungsi Lambda, lakukan hal berikut:
 - a. [Untuk Menggunakan integrasi proxy Lambda, pilih kotak centang jika Anda ingin menggunakan integrasi proxy Lambda atau integrasi proxy Lambda lintas akun.](#)
 - b. Untuk fungsi Lambda, tentukan fungsi dengan salah satu cara berikut:
 - Jika fungsi Lambda Anda berada di akun yang sama, masukkan nama fungsi dan kemudian pilih fungsi dari daftar dropdown.

 Note

Nama fungsi secara opsional dapat menyertakan alias atau spesifikasi versinya, seperti dalam, HelloWorldHelloWorld:1, atau. HelloWorld:alpha

- Jika fungsinya ada di akun yang berbeda, masukkan ARN untuk fungsi tersebut.
- c. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000

6. Jika Anda memilih HTTP, ikuti petunjuk di langkah 4 dari [the section called “Siapkan permintaan integrasi menggunakan konsol”](#).
7. Jika Anda memilih Mock, lanjutkan ke langkah Permintaan Template.
8. Jika Anda memilih AWS layanan, ikuti instruksi pada langkah 6 dari [the section called “Siapkan permintaan integrasi menggunakan konsol”](#).
9. Jika Anda memilih tautan VPC, lakukan hal berikut:
 - a. Untuk integrasi proxy VPC, pilih kotak centang jika Anda ingin permintaan Anda diproksi ke titik akhir Anda. VPCLink
 - b. Untuk metode HTTP, pilih jenis metode HTTP yang paling cocok dengan metode di backend HTTP.
 - c. Dari daftar dropdown tautan VPC, pilih tautan VPC. Anda dapat memilih [Use Stage Variables] dan memasukkan `${stageVariables.vpcLinkId}` dalam kotak teks di bawah daftar.

Anda dapat menentukan variabel vpcLinkId stage setelah menerapkan API ke tahap dan menetapkan nilainya ke ID. VpcLink
 - d. Untuk URL Endpoint, masukkan URL backend HTTP yang ingin digunakan integrasi ini.
 - e. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara 50 dan milidetik. 29000
10. Pilih Simpan perubahan.
11. Di bawah templat Permintaan, lakukan hal berikut:
 - a. Untuk memasukkan ekspresi pemilihan Template, di bawah Permintaan template, pilih Edit.
 - b. Masukkan ekspresi pemilihan Template. Gunakan ekspresi yang dicari API Gateway di payload pesan. Jika ditemukan, itu dievaluasi, dan hasilnya adalah nilai kunci template yang digunakan untuk memilih template pemetaan data yang akan diterapkan ke data dalam payload pesan. Anda membuat template pemetaan data di langkah berikutnya. Pilih Edit untuk menyimpan perubahan Anda.
 - c. Pilih Buat template untuk membuat template pemetaan data. Untuk kunci Template, masukkan nilai kunci template yang digunakan untuk memilih template pemetaan data yang akan diterapkan ke data dalam payload pesan. Kemudian, masukkan template pemetaan. Pilih Buat templat.

Untuk informasi tentang ekspresi pemilihan templat, lihat [the section called “Ekspresi pemilihan template”](#).

Siapkan permintaan integrasi menggunakan AWS CLI

Anda dapat menyiapkan permintaan integrasi untuk rute di WebSocket API dengan menggunakan AWS CLI seperti pada contoh berikut, yang membuat integrasi tiruan:

1. Buat file bernama `integration-params.json`, dengan konten berikut:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
  "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
    "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Gunakan perintah [create-integration](#) berikut untuk membuat integrasi tiruan.

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccdde --cli-
input-json file://integration-params.json
```

Outputnya akan terlihat seperti berikut:

```
{
  "PassthroughBehavior": "WHEN_NO_MATCH",
  "TimeoutInMillis": 29000,
  "ConnectionType": "INTERNET",
  "IntegrationResponseSelectionExpression": "${response.statuscode}",
  "RequestTemplates": {
    "application/json": "{\"statusCode\":200}"
  },
  "IntegrationId": "0abcdef",
  "IntegrationType": "MOCK"
}
```

Atau, Anda dapat mengatur permintaan integrasi untuk integrasi proxy dengan menggunakan AWS CLI.

1. Buat fungsi Lambda di konsol Lambda dan berikan peran eksekusi Lambda dasar.
2. Gunakan perintah [create-integration](#) berikut untuk membuat integrasi.

```
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
```

Outputnya akan terlihat seperti berikut:

```
{
  "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"
}
```

Format input fungsi Lambda untuk integrasi proxy untuk WebSocket APIs

Dalam integrasi proxy Lambda, API Gateway memetakan seluruh permintaan klien ke event parameter input fungsi Lambda backend. Contoh berikut menunjukkan struktur peristiwa masukan dari \$connect rute dan peristiwa masukan dari rute yang dikirimkan API Gateway ke integrasi proxy Lambda. \$disconnect

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
}
```

Menyiapkan respons integrasi WebSocket API di API Gateway

Bagian berikut memberikan gambaran singkat tentang respons integrasi untuk WebSocket API dan cara menyiapkan respons integrasi untuk WebSocket API.

Topik

- [Ikhtisar tanggapan integrasi](#)
- [Respons integrasi untuk komunikasi dua arah](#)
- [Menyiapkan respons integrasi menggunakan konsol API Gateway](#)
- [Siapkan respons integrasi menggunakan AWS CLI](#)

Ikhtisar tanggapan integrasi

Respons integrasi API Gateway adalah cara memodelkan dan memanipulasi respons dari layanan backend. Ada beberapa perbedaan dalam penyiapan REST API versus respons integrasi WebSocket API, tetapi secara konseptual perilakunya sama.

WebSocket rute dapat dikonfigurasi untuk komunikasi dua arah atau satu arah.

- Saat rute dikonfigurasi untuk komunikasi dua arah, respons integrasi memungkinkan Anda mengonfigurasi transformasi pada muatan pesan yang dikembalikan, mirip dengan respons integrasi untuk REST APIs
- Jika rute dikonfigurasi untuk komunikasi satu arah, maka terlepas dari konfigurasi respons integrasi apa pun, tidak ada respons yang akan dikembalikan melalui WebSocket saluran setelah pesan diproses.

API Gateway tidak akan meneruskan respons backend ke respons rute, kecuali jika Anda menyiapkan respons rute. Untuk mempelajari cara menyiapkan respons rute, lihat [the section called “Siapkan respons rute WebSocket API”](#).

Respons integrasi untuk komunikasi dua arah

Integrasi dapat dibagi menjadi integrasi proxy dan integrasi non-proxy.

 Important

Untuk integrasi proxy, API Gateway secara otomatis meneruskan output backend ke pemanggil sebagai muatan lengkap. Tidak ada respon integrasi.

Untuk integrasi non-proxy, Anda harus menyiapkan setidaknya satu respons integrasi:

- Idealnya, salah satu tanggapan integrasi Anda harus bertindak sebagai tangkapan semua ketika tidak ada pilihan eksplisit yang dapat dibuat. Kasus default ini diwakili dengan menyetel kunci respons integrasi dari `$default`.
- Dalam semua kasus lain, kunci respons integrasi berfungsi sebagai ekspresi reguler. Ini harus mengikuti format `/expression/`.

Untuk integrasi HTTP non-proxy:

- API Gateway akan mencoba mencocokkan kode status HTTP dari respons backend. Tombol respons integrasi akan berfungsi sebagai ekspresi reguler dalam kasus ini. Jika kecocokan tidak dapat ditemukan, `$default` maka dipilih sebagai respons integrasi.
- Ekspresi pemilihan template, seperti dijelaskan di atas, berfungsi secara identik. Contoh:
 - `/2\d\d/`: Menerima dan mengubah tanggapan yang berhasil
 - `/4\d\d/`: Menerima dan mengubah kesalahan permintaan yang buruk

- `$default`: Menerima dan mengubah semua tanggapan tak terduga

Untuk informasi selengkapnya tentang ekspresi pemilihan templat, lihat [the section called “Ekspresi pemilihan template”](#).

Menyiapkan respons integrasi menggunakan konsol API Gateway

Untuk menyiapkan respons integrasi rute untuk WebSocket API menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih WebSocket API Anda dan pilih rute Anda.
3. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan respons integrasi, pilih Buat respons integrasi.
4. Untuk tombol Respons, masukkan nilai yang akan ditemukan di kunci respons dalam pesan keluar setelah mengevaluasi ekspresi pemilihan respons. Misalnya, Anda dapat memasukkan `/4\d\d/` untuk menerima dan mengubah kesalahan permintaan buruk atau masuk `$default` untuk menerima dan mengubah semua respons yang cocok dengan ekspresi pemilihan templat.
5. Untuk ekspresi pemilihan Template, masukkan ekspresi seleksi untuk mengevaluasi pesan keluar.
6. Pilih Buat respons.
7. Anda juga dapat menentukan template pemetaan untuk mengonfigurasi transformasi payload pesan yang dikembalikan. Pilih Buat templat.
8. Masukkan nama kunci. Jika Anda memilih ekspresi pemilihan template default, masukkan `\$default`.
9. Untuk template Response, masukkan template pemetaan Anda di editor kode.
10. Pilih Buat templat.
11. Pilih Deploy API untuk menerapkan API Anda.

Gunakan perintah [wscat](#) berikut untuk terhubung ke API Anda. Untuk informasi selengkapnya tentang `wscat`, lihat [the section called “Gunakan wscat untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Saat Anda memanggil rute Anda, muatan pesan yang dikembalikan akan kembali.

Siapkan respons integrasi menggunakan AWS CLI

[create-integration-response](#) Perintah berikut menciptakan respons `$default` integrasi:

```
aws apigatewayv2 create-integration-response \  
  --api-id vaz7da96z6 \  
  --integration-id a1b2c3 \  
  --integration-response-key '$default'
```

Minta validasi untuk WebSocket APIs di API Gateway

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi pada permintaan rute sebelum melanjutkan dengan permintaan integrasi. Jika validasi gagal, API Gateway gagal permintaan tanpa memanggil backend Anda, mengirimkan respons gateway “Badan permintaan buruk” ke klien, dan menerbitkan hasil validasi di Log. CloudWatch Menggunakan validasi dengan cara ini mengurangi panggilan yang tidak perlu ke backend API Anda.

Ekspresi pemilihan model

Anda dapat menggunakan ekspresi pemilihan model untuk memvalidasi permintaan secara dinamis dalam rute yang sama. Validasi model terjadi jika Anda memberikan ekspresi pemilihan model untuk integrasi proxy atau non-proxy. Anda mungkin perlu mendefinisikan `$default` model sebagai fallback ketika tidak ada model yang cocok ditemukan. Jika tidak ada model yang cocok dan `$default` tidak ditentukan, validasi gagal. Ekspresi seleksi terlihat seperti `Route.ModelSelectionExpression` dan mengevaluasi kunci untuk `Route.RequestModels`.

Saat menentukan rute untuk WebSocket API, Anda dapat menentukan ekspresi pemilihan model secara opsional. Ekspresi ini dievaluasi untuk memilih model yang akan digunakan untuk validasi tubuh ketika permintaan diterima. Ekspresi mengevaluasi ke salah satu entri dalam rute.

[requestmodels](#)

Sebuah model dinyatakan sebagai [skema JSON](#) dan menggambarkan struktur data dari badan permintaan. Sifat ekspresi seleksi ini memungkinkan Anda memilih model secara dinamis untuk divalidasi saat runtime untuk rute tertentu. Untuk informasi tentang cara membuat model, lihat [the section called “Model data untuk REST APIs”](#).

Menyiapkan validasi permintaan menggunakan konsol API Gateway

Contoh berikut menunjukkan cara mengatur validasi permintaan pada rute.

Pertama, Anda membuat model, dan kemudian Anda membuat rute. Selanjutnya, Anda mengonfigurasi validasi permintaan pada rute yang baru saja Anda buat. Terakhir, Anda menerapkan dan menguji API Anda. Untuk menyelesaikan tutorial ini, Anda memerlukan WebSocket API dengan `$request.body.action` ekspresi pemilihan rute dan titik akhir integrasi untuk rute baru Anda.

Anda juga `wscat` perlu terhubung ke API Anda. Untuk informasi selengkapnya, lihat [the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).

Untuk membuat model

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih WebSocket API.
3. Di panel navigasi utama, pilih Model.
4. Pilih Buat model.
5. Untuk Nama, masukkan **emailModel**.
6. Untuk jenis Konten, masukkan **application/json**.
7. Untuk skema Model, masukkan model berikut:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type" : "object",
  "required" : [ "address"],
  "properties" : {
    "address": {
      "type": "string"
    }
  }
}
```

Model ini mengharuskan permintaan berisi alamat email.

8. Pilih Simpan.

Pada langkah ini, Anda membuat rute untuk WebSocket API Anda.

Untuk membuat rute

1. Di panel navigasi utama, pilih Rute.
2. Pilih Buat rute.

3. Untuk kunci Rute, masukkan **sendMessage**.
4. Pilih jenis integrasi dan tentukan titik akhir integrasi. Untuk mengetahui informasi selengkapnya, lihat [the section called “Integrasi”](#).
5. Pilih Buat rute.

Pada langkah ini, Anda menyiapkan validasi permintaan untuk sendMessage rute tersebut.

Untuk mengatur validasi permintaan

1. Pada tab Permintaan rute, di bawah Pengaturan permintaan rute, pilih Edit.
2. Untuk ekspresi pemilihan Model, masukkan **`${request.body.messageType}`**.

API Gateway menggunakan messageType properti untuk memvalidasi permintaan yang masuk.

3. Pilih Tambahkan model permintaan.
4. Untuk kunci Model, masukkan **email**.
5. Untuk Model, pilih EmailModel.

API Gateway memvalidasi pesan masuk dengan messageType properti yang disetel ke email terhadap model ini.

 Note

Jika API Gateway tidak dapat mencocokkan ekspresi pemilihan model dengan kunci model, maka ia memilih \$default model. Jika tidak ada \$default model, maka validasi gagal. Untuk produksi APIs, kami sarankan Anda membuat \$default model.

6. Pilih Simpan perubahan.

Pada langkah ini, Anda menerapkan dan menguji API Anda.

Untuk menerapkan dan menguji API Anda

1. Pilih Deploy API.
2. Pilih tahap yang diinginkan dari daftar dropdown atau masukkan nama tahap baru.
3. Pilih Deploy.
4. Di panel navigasi utama, pilih Tahapan.

5. Salin WebSocket URL API Anda. URL akan terlihat seperti `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Buka terminal baru dan jalankan `wscat` perintah dengan parameter berikut.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Gunakan perintah berikut untuk menguji API Anda.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId": "ABCD1=234",  
  "requestId": "EFGH="}
```

API Gateway akan gagal permintaan.

Gunakan perintah berikutnya untuk mengirim permintaan yang valid ke API Anda.

```
{"action": "sendMessage", "messageType": "email", "address":  
  "mary_major@example.com"}
```

Transformasi data untuk WebSocket APIs di API Gateway

Di API Gateway, permintaan metode WebSocket API dapat mengambil payload dalam format yang berbeda dari payload permintaan integrasi yang sesuai, seperti yang diperlukan di backend. Demikian pula, backend dapat mengembalikan payload respons integrasi yang berbeda dari payload respons metode, seperti yang diharapkan oleh frontend.

API Gateway memungkinkan Anda menggunakan transformasi template pemetaan untuk memetakan payload dari permintaan metode ke permintaan integrasi yang sesuai dan dari respons integrasi ke respons metode yang sesuai. Anda membuat template pemetaan dan Anda menentukan ekspresi pemilihan template untuk menentukan template mana yang akan digunakan untuk melakukan transformasi data yang diperlukan.

Anda dapat menggunakan pemetaan data untuk memetakan data dari [permintaan rute ke integrasi backend](#). Untuk mempelajari selengkapnya, lihat [the section called “Pemetaan data”](#).

Memetakan template dan model

Template pemetaan [adalah skrip yang dinyatakan dalam Velocity Template Language \(VTL\) dan diterapkan pada payload menggunakan ekspresi. JSONPath](#). Untuk informasi selengkapnya tentang template pemetaan API Gateway, lihat [Memetakan transformasi template untuk REST API di API Gateway](#).

Muatan dapat memiliki model data sesuai dengan rancangan [skema JSON 4](#). Anda tidak perlu mendefinisikan model untuk membuat template pemetaan. Namun, model dapat membantu Anda membuat template karena API Gateway menghasilkan cetak biru template berdasarkan model yang disediakan. Untuk informasi selengkapnya tentang model API Gateway, lihat [Model data untuk REST APIs](#).

Ekspresi pemilihan template

Untuk mengubah payload dengan template pemetaan, Anda menentukan ekspresi pemilihan template WebSocket API dalam [permintaan integrasi atau respons integrasi](#). Ekspresi ini dievaluasi untuk menentukan template input atau output (jika ada) yang akan digunakan untuk mengubah badan permintaan menjadi badan permintaan integrasi (melalui template input) atau badan respons integrasi ke badan respons rute (melalui template keluaran).

`Integration.TemplateSelectionExpression` mendukung `${request.body.jsonPath}` dan nilai statis.

`IntegrationResponse.TemplateSelectionExpression` mendukung `${request.body.jsonPath}`, dan nilai-nilai statis.

Ekspresi pemilihan respons integrasi

Saat [menyiapkan respons integrasi](#) untuk WebSocket API, Anda dapat menentukan ekspresi pemilihan respons integrasi secara opsional. Ekspresi ini menentukan apa yang [IntegrationResponse](#) harus dipilih ketika integrasi kembali. Nilai ekspresi ini saat ini dibatasi oleh API Gateway, seperti yang didefinisikan di bawah ini. Sadarilah bahwa ekspresi ini hanya relevan untuk integrasi non-proxy; integrasi proxy hanya meneruskan payload respons kembali ke pemanggil tanpa pemodelan atau modifikasi.

Tidak seperti ekspresi seleksi sebelumnya lainnya, ekspresi ini saat ini mendukung format pencocokan pola. Ekspresi harus dibungkus dengan garis miring ke depan.

Saat ini nilainya tetap tergantung pada [integrationType](#):

- Untuk integrasi berbasis Lambda, memang demikian.
`$integration.response.body.errorMessage`
- Untuk HTTP dan MOCK integrasi, itu `$integration.response.statuscode`.
- Untuk HTTP_PROXY dan AWS_PROXY, ungkapan tidak digunakan karena Anda meminta agar payload diteruskan ke penelepon.

Mengatur pemetaan data WebSocket APIs di API Gateway

Pemetaan data memungkinkan Anda memetakan data dari [permintaan rute ke integrasi](#) backend.

Note

Pemetaan data untuk WebSocket APIs tidak didukung di. Konsol Manajemen AWS Anda harus menggunakan AWS CLI, AWS CloudFormation, atau SDK untuk mengonfigurasi pemetaan data.

Topik

- [Petakan data permintaan rute ke parameter permintaan integrasi](#)
- [Contoh](#)

Petakan data permintaan rute ke parameter permintaan integrasi

Parameter permintaan integrasi dapat dipetakan dari parameter permintaan rute yang ditentukan, badan permintaan, [context](#) atau [stage](#) variabel, dan nilai statis.

Tabel berikut menunjukkan ekspresi pemetaan data permintaan integrasi. Dalam tabel, *PARAM_NAME* adalah nama parameter permintaan rute dari jenis parameter yang diberikan. Itu harus cocok dengan ekspresi reguler `^[a-zA-Z0-9._$-]+$`. *JSONPath_EXPRESSION* adalah JSONPath ekspresi untuk bidang JSON dari badan permintaan.

Sumber data yang dipetakan	Ekspresi pemetaan
Minta string kueri (hanya didukung untuk \$connect rute)	<code>route.request.querystring. <i>PARAM_NAME</i></code>

Sumber data yang dipetakan	Ekspresi pemetaan
Permintaan header (didukung hanya untuk \$connect rute)	<code>route.request.header. <i>PARAM_NAME</i></code>
String kueri permintaan multi-nilai (hanya didukung untuk \$connect rute)	<code>route.request.multivaluequerystring. <i>PARAM_NAME</i></code>
Header permintaan multi-nilai (hanya didukung untuk \$connect rute)	<code>route.request.multivalueheader. <i>PARAM_NAME</i></code>
Isi permintaan	<code>route.request.body. <i>JSONPath_EXPRESSION</i></code>
Variabel tahap	<code>stageVariables. <i>VARIABLE_NAME</i></code>
Variabel konteks	<code>context.<i>VARIABLE_NAME</i></code> yang harus menjadi salah satu variabel konteks yang didukung .
Nilai statis	<code>'<i>STATIC_VALUE</i>' . <i>STATIC_VALUE</i></code> Ini adalah string literal dan harus diapit dalam tanda kutip tunggal.

Saat Anda membuat pemetaan data, gunakan AWS CLI pastikan untuk mengikuti format yang benar untuk menggunakan literal dengan string di. AWS CLI Untuk informasi selengkapnya, lihat [Menggunakan tanda kutip dan literal dengan string AWS CLI di Panduan Pengguna](#). AWS Command Line Interface

Contoh

AWS CLI Contoh berikut mengkonfigurasi pemetaan data. Untuk contoh CloudFormation template, lihat [websocket-data-mapping.yaml](#).

Memetakan ConnectionId klien ke header dalam permintaan integrasi

Perintah [update-integration](#) berikut memetakan klien connectionId ke connectionId header dalam permintaan ke integrasi backend:

```
aws apigatewayv2 update-integration \
```

```
--integration-id abc123 \  
--api-id a1b2c3d4 \  
--request-parameters  
'integration.request.header.connectionId'='context.connectionId'
```

Petakan parameter string kueri ke header dalam permintaan integrasi

Contoh berikut memetakan parameter string authToken query ke authToken header dalam permintaan integrasi.

1. Gunakan perintah [update-route](#) berikut untuk menambahkan parameter string authToken kueri ke parameter permintaan rute.

```
aws apigatewayv2 update-route --route-id 0abcdef \  
--api-id a1b2c3d4 \  
--request-parameters '{"route.request.querystring.authToken": {"Required":  
false}}'
```

2. Gunakan perintah [update-integration](#) berikut untuk memetakan parameter string query ke authToken header dalam permintaan untuk integrasi backend.

```
aws apigatewayv2 update-integration \  
--integration-id abc123 \  
--api-id a1b2c3d4 \  
--request-parameters  
'integration.request.header.authToken'='route.request.querystring.authToken'
```

3. (Opsional) Jika perlu, gunakan yang berikut ini [delete-route-request-parameter](#) untuk menghapus parameter string authToken kueri dari parameter permintaan rute.

```
aws apigatewayv2 delete-route-request-parameter \  
--route-id 0abcdef \  
--api-id a1b2c3d4 \  
--request-parameter-key 'route.request.querystring.authToken'
```

WebSocket Referensi template pemetaan API untuk API Gateway

Bagian ini merangkum kumpulan variabel yang saat ini didukung WebSocket APIs di API Gateway.

Parameter	Deskripsi
<code>\$context.connectionId</code>	ID unik untuk koneksi yang dapat digunakan untuk membuat callback ke klien.
<code>\$context.connectedAt</code>	Waktu koneksi yang diformat Epoch .
<code>\$context.domainName</code>	Nama domain untuk WebSocket API. Ini dapat digunakan untuk membuat callback ke klien (bukan nilai hard-code).
<code>\$context.eventType</code>	Jenis acara:CONNECT,MESSAGE, atauDISCONNECT .
<code>\$context.messageId</code>	ID sisi server unik untuk pesan. Hanya tersedia ketika <code>\$context.eventType</code> adaMESSAGE.
<code>\$context.routeKey</code>	Kunci rute yang dipilih.
<code>\$context.requestId</code>	Sama seperti <code>\$context.extendedRequestId</code> .
<code>\$context.extendedRequestId</code>	ID yang dibuat secara otomatis untuk panggilan API, yang berisi informasi yang lebih berguna untuk debugging/pemecahan masalah.
<code>\$context.apiId</code>	API Gateway identifier ditetapkan ke API Anda.
<code>\$context.authorizer.principalId</code>	Identifikasi pengguna utama yang terkait dengan token yang dikirim oleh klien dan dikembalikan dari fungsi Lambda API Gateway (sebelumnya dikenal sebagai otorisasi khusus).
<code>\$context.authorizer. <i>property</i></code>	Nilai stringifikasi dari pasangan nilai kunci context peta yang ditentukan dikembalikan dari fungsi otorisasi API Gateway Lambda.

Parameter	Deskripsi
	Misalnya, jika otorisasi mengembalikan context peta berikut: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> memanggil <code>\$context.authorizer.key</code> mengembalikan "value" string, memanggil <code>\$context.authorizer.numKey</code> mengembalikan "1" string, dan memanggil <code>\$context.authorizer.boolKey</code> mengembalikan "true" string.
<code>\$context.error.messageString</code>	Nilai yang dikutip dari <code>\$context.error.message</code> , yaitu <code>"\$context.error.message"</code> .
<code>\$context.error.validationErrorString</code>	Sebuah string yang berisi pesan kesalahan validasi rinci.
<code>\$context.identity.accountId</code>	ID AWS akun yang terkait dengan permintaan.
<code>\$context.identity.apiKey</code>	Kunci pemilik API yang terkait dengan permintaan API berkemampuan kunci.
<code>\$context.identity.apiKeyId</code>	ID kunci API yang terkait dengan permintaan API berkemampuan kunci
<code>\$context.identity.caller</code>	Pengidentifikasi utama penelepon yang membuat permintaan.

Parameter	Deskripsi
<code>\$context.identity.cognitoAuthenticationProvider</code>	Daftar dipisahkan koma dari semua penyedia otentikasi Amazon Cognito yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Untuk informasi tentang penyedia autentikasi Amazon Cognito yang tersedia, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Nilai yang mungkin termasuk <code>authenticated</code> untuk identitas yang diautentikasi dan <code>unauthenticated</code> untuk identitas yang tidak diautentikasi.
<code>\$context.identity.cognitoIdentityId</code>	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.

Parameter	Deskripsi
<code>\$context.identity.sourceIp</code>	Alamat IP sumber koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
<code>\$context.identity.user</code>	Pengidentifikasi utama pengguna yang membuat permintaan.
<code>\$context.identity.userAgent</code>	Agen Pengguna pemanggil API.
<code>\$context.identity.userArn</code>	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi.
<code>\$context.requestTime</code>	Waktu permintaan yang diformat CLF (). dd/MMM/yyyy:HH:mm:ss +-hhmm
<code>\$context.requestTimeEpoch</code>	Waktu permintaan yang diformat Epoch , dalam milidetik.
<code>\$context.stage</code>	Tahap penerapan panggilan API (misalnya, Beta atau Prod).
<code>\$context.status</code>	Status respons.
<code>\$input.body</code>	Mengembalikan payload mentah sebagai string.
<code>\$input.json(x)</code>	Fungsi ini mengevaluasi JSONPath ekspresi dan mengembalikan hasil sebagai string JSON. Misalnya, <code>\$input.json('\$.pets')</code> akan mengembalikan string JSON yang mewakili struktur hewan peliharaan. Untuk informasi selengkapnya tentang JSONPath, lihat JSONPath atau JSONPath untuk Java.

Parameter	Deskripsi
<code>\$input.path(x)</code>	Mengambil JSONPath ekspresi string (x) dan mengembalikan representasi objek JSON dari hasil. Ini memungkinkan Anda untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika ekspresi <code>\$input.path('\$.pets')</code> mengembalikan objek seperti ini: <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> akan kembali "3". Untuk informasi selengkapnya tentang JSONPath, lihat JSONPath atau JSONPath untuk Java.
<code>\$stageVariables. <variable_name></code>	<code><variable_name></code> merupakan nama variabel tahap.

Parameter	Deskripsi
<code>\$stageVariables[' <variable_name> ']</code>	<code><variable_name></code> mewakili setiap nama variabel tahap.
<code>\${stageVariables[' <variable_name>']}</code>	<code><variable_name></code> mewakili setiap nama variabel tahap.
<code>\$util.escapeJavaScript()</code>	Melarikan diri dari karakter dalam string menggunakan aturan JavaScript string.  Note Fungsi ini akan mengubah tanda kutip tunggal biasa (') menjadi yang keluar (\'). Namun, tanda kutip tunggal yang lolos tidak valid di JSON. Jadi, ketika output dari fungsi ini digunakan dalam properti JSON, Anda harus mengubah tanda kutip tunggal yang diloloskan (\') kembali ke tanda kutip tunggal biasa ('). Ini ditunjukkan dalam contoh berikut: <pre>\$util.escapeJavaScript(ript(<i>data</i>).replaceAll("\\'", ,"'")</pre>

Parameter	Deskripsi
<code>\$util.parseJson()</code>	Mengambil “stringified” JSON dan mengembalikan representasi objek dari hasilnya. Anda dapat menggunakan hasil dari fungsi ini untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika Anda memiliki muatan berikut: <pre>{ "errorMessage": "{ \"key1\": \"var1\", \"key2\": { \"arr\": [1,2,3] } }" }</pre> dan gunakan template pemetaan berikut <pre>#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$.errorMessage'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> Anda akan mendapatkan output sebagai berikut: <pre>{ "errorMessageObjKey2ArrVal" : 1 }</pre>
<code>\$util.urlEncode()</code>	Mengkonversi string ke dalam format “aplikasi/x-www-form-urlencoded”.
<code>\$util.urlDecode()</code>	Mendekode string “aplikasi/x-www-form-urlencoded”.
<code>\$util.base64Encode()</code>	Mengkodekan data ke dalam string yang dikodekan base64.

Parameter	Deskripsi
<code>\$util.base64Decode()</code>	Mendekode data dari string yang dikodekan base64.

Jenis media biner untuk WebSocket APIs di API Gateway

API Gateway saat ini WebSocket APIs tidak mendukung bingkai biner dalam muatan pesan masuk. Jika aplikasi klien mengirimkan bingkai biner, API Gateway menolaknya dan memutus klien dengan kode 1003.

Ada solusi untuk perilaku ini. Jika klien mengirimkan data biner yang disandikan teks (misalnya, base64) sebagai bingkai teks, Anda dapat mengatur `contentHandlingStrategy` properti integrasi untuk mengonversi muatan dari string yang dikodekan base64 `CONVERT_TO_BINARY` ke biner.

Untuk mengembalikan respons rute untuk muatan biner dalam integrasi non-proxy, Anda dapat menyetel `contentHandlingStrategy` properti respons integrasi untuk mengonversi payload dari string biner `CONVERT_TO_TEXT` ke string berenkode base64.

Memohon WebSocket APIs

Setelah menerapkan WebSocket API, aplikasi klien dapat terhubung dan mengirim pesan ke API itu —dan layanan backend Anda dapat mengirim pesan ke aplikasi klien yang terhubung:

- Anda dapat menggunakan `wscat` untuk menyambung ke WebSocket API Anda dan mengirim pesan ke sana untuk mensimulasikan perilaku klien. Lihat [the section called “Gunakan wscat untuk terhubung ke WebSocket API dan mengirim pesan ke sana”](#).
- Anda dapat menggunakan `@connections` API dari layanan backend Anda untuk mengirim pesan callback ke klien yang terhubung, mendapatkan informasi koneksi, atau memutuskan sambungan klien. Lihat [the section called “Gunakan @connections perintah di layanan backend Anda”](#).
- Aplikasi klien dapat menggunakan WebSocket pustaka sendiri untuk menjalankan WebSocket API Anda.

Gunakan **wscat** untuk terhubung ke WebSocket API dan mengirim pesan ke sana

[wscat](#) Utilitas adalah alat yang nyaman untuk menguji WebSocket API yang telah Anda buat dan gunakan di API Gateway. Anda dapat menginstal dan menggunakan `wscat` sebagai berikut:

1. Unduh wscat dari <https://www.npmjs.com/package/wscat>.
2. Instal wscat dengan menjalankan perintah berikut:

```
npm install -g wscat
```

3. Untuk terhubung ke API Anda, jalankan wscat perintah seperti yang ditunjukkan pada contoh berikut. Perhatikan bahwa contoh ini mengasumsikan bahwa Authorization pengaturannya adalah NONE.

```
wscat -c wss://aabbccdde.execute-api.us-east-1.amazonaws.com/test/
```

Anda perlu mengganti *aabbccdde* dengan ID API yang sebenarnya, yang ditampilkan di konsol API Gateway atau dikembalikan oleh AWS CLI [create-api](#) perintah.

Selain itu, jika API Anda berada di Wilayah selain *us-east-1*, Anda perlu mengganti Region yang benar.

4. Untuk menguji API Anda, masukkan pesan seperti berikut saat terhubung:

```
{"jsonpath-expression":"route-key"}
```

di mana *jsonpath-expression* adalah JSONPath ekspresi dan *route-key* merupakan kunci rute untuk API. Contoh:

```
{"action":"action1"}  
{"message":"test response body"}
```

Untuk informasi selengkapnya tentang JSONPath, lihat [JSONPath](#) atau [JSONPath untuk Java](#).

5. Untuk memutuskan sambungan dari API Anda, masukkan `ctrl-C`.

Gunakan **@connections** perintah di layanan backend Anda

Layanan backend Anda dapat menggunakan permintaan HTTP WebSocket koneksi berikut untuk mengirim pesan callback ke klien yang terhubung, mendapatkan informasi koneksi, atau memutuskan sambungan klien.

⚠ Important

Permintaan ini menggunakan [otorisasi IAM](#), jadi Anda harus menandatangani dengan [Sigv4 \(SigV4\)](#). Untuk melakukan ini, Anda dapat menggunakan API Gateway Management API. Untuk informasi selengkapnya, lihat [ApiGatewayManagementApi](#).

Pada perintah berikut, Anda perlu mengganti `{api-id}` dengan ID API yang sebenarnya, yang ditampilkan di konsol API Gateway atau dikembalikan oleh perintah AWS CLI [create-api](#). Anda harus membuat koneksi sebelum menggunakan perintah ini.

Untuk mengirim pesan callback ke klien, gunakan:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Anda dapat menguji permintaan ini dengan menggunakan [Postman](#) atau dengan memanggil [awscurl](#) seperti pada contoh berikut:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Anda perlu mengkodekan URL perintah seperti pada contoh berikut:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccdde.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Untuk mendapatkan status koneksi terbaru klien, gunakan:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Untuk memutuskan sambungan klien, gunakan:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Anda dapat membuat URL callback secara dinamis dengan menggunakan `$context` variabel dalam integrasi Anda. Misalnya, jika Anda menggunakan integrasi proxy Lambda dengan fungsi `Node.js` Lambda, Anda dapat membuat URL dan mengirim pesan ke klien yang terhubung sebagai berikut:

```
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,
  };
};
```

Jika Anda menggunakan nama domain khusus untuk WebSocket API Anda, hapus `stage` variabel dari kode fungsi Anda.

Saat mengirim pesan panggilan balik, fungsi Lambda Anda harus memiliki izin untuk memanggil API Gateway Management API. Anda mungkin menerima kesalahan yang berisi `GoneException` jika Anda memposting pesan sebelum koneksi dibuat, atau setelah klien terputus.

Publikasikan WebSocket APIs untuk pelanggan untuk dipanggil

Cukup membuat dan mengembangkan API Gateway API tidak secara otomatis membuatnya dapat dipanggil oleh pengguna Anda. Untuk membuatnya dapat dipanggil, Anda harus menerapkan API Anda ke sebuah panggung. Selain itu, Anda mungkin ingin menyesuaikan URL yang akan digunakan pengguna Anda untuk mengakses API Anda. Anda dapat memberikan domain yang konsisten dengan merek Anda atau lebih mudah diingat daripada URL default untuk API Anda.

Di bagian ini, Anda dapat mempelajari cara menerapkan API dan menyesuaikan URL yang Anda berikan kepada pengguna untuk mengaksesnya.

Note

Untuk meningkatkan keamanan API Gateway Anda APIs, `execute-api`. `{region}.amazonaws.com` domain terdaftar di [Daftar Akhiran Publik \(PSL\)](#). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway APIs Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman [Set-Cookie](#) di Jaringan Pengembang Mozilla.

Topik

- [Buat tahapan untuk WebSocket APIs di API Gateway](#)
- [Terapkan WebSocket APIs di API Gateway](#)
- [Kebijakan keamanan untuk WebSocket APIs di API Gateway](#)
- [Nama domain khusus untuk WebSocket APIs di API Gateway](#)

Buat tahapan untuk WebSocket APIs di API Gateway

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,, `dev` `prodbeta`, atau `v2`). Tahapan API diidentifikasi oleh ID API dan nama stage mereka, dan mereka disertakan dalam URL yang Anda gunakan untuk memanggil API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

Deployment adalah snapshot dari konfigurasi API Anda. Setelah Anda menerapkan API ke tahap, itu tersedia bagi klien untuk dipanggil. Anda harus menerapkan API agar perubahan diterapkan.

Variabel tahap

Variabel tahap adalah pasangan nilai kunci yang dapat Anda tentukan untuk tahap API. WebSocket Mereka bertindak seperti variabel lingkungan dan dapat digunakan dalam pengaturan API Anda.

Misalnya, Anda dapat menentukan variabel tahap, dan kemudian menetapkan nilainya sebagai titik akhir HTTP untuk integrasi proxy HTTP. Kemudian, Anda dapat mereferensikan titik akhir dengan menggunakan nama variabel tahap terkait. Dengan melakukan ini, Anda dapat menggunakan penyiapan API yang sama dengan titik akhir yang berbeda di setiap tahap. Demikian pula, Anda dapat menggunakan variabel tahap untuk menentukan integrasi AWS Lambda fungsi yang berbeda untuk setiap tahap API Anda.

Note

Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensial. Untuk meneruskan data sensitif ke integrasi, gunakan AWS Lambda otorisasi. Anda dapat meneruskan data sensitif ke integrasi dalam output otorisasi Lambda. Untuk mempelajari selengkapnya, lihat [the section called “Format respons otorisasi Lambda”](#).

Contoh

Untuk menggunakan variabel tahap untuk menyesuaikan titik akhir integrasi HTTP, Anda harus terlebih dahulu mengatur nama dan nilai variabel tahap (misalnya, `url`) dengan `nilaiexample.com`. Selanjutnya, siapkan integrasi proxy HTTP. Alih-alih memasukkan URL titik akhir, Anda dapat memberi tahu API Gateway untuk menggunakan nilai variabel stage, **`http://${stageVariables.url}`**. Nilai ini memberi tahu API Gateway untuk mengganti variabel stage Anda `${}` saat runtime, tergantung pada tahap API Anda.

Anda dapat mereferensikan variabel tahap dengan cara yang sama untuk menentukan nama fungsi Lambda atau peran AWS ARN.

Saat menentukan nama fungsi Lambda sebagai nilai variabel tahap, Anda harus mengonfigurasi izin pada fungsi Lambda secara manual. Perintah [add-permission](#) berikut menambahkan izin yang diperlukan:

```
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
```

Referensi variabel tahap API Gateway

Integrasi HTTP URIs

Anda dapat menggunakan variabel tahap sebagai bagian dari URI integrasi HTTP, seperti yang ditunjukkan pada contoh berikut.

- URI lengkap tanpa protokol — `http://${stageVariables.<variable_name>}`
- Domain lengkap — `http://${stageVariables.<variable_name>}/resource/operation`
- Sebuah subdomain — `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Sebuah jalan — `http://example.com/${stageVariables.<variable_name>}/bar`
- Sebuah string kueri - `http://example.com/foo?q=${stageVariables.<variable_name>}`

Fungsi Lambda

Anda dapat menggunakan variabel tahap di tempat nama fungsi Lambda atau alias, seperti yang ditunjukkan dalam contoh berikut.

- `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

Untuk menggunakan variabel stage untuk fungsi Lambda, fungsi tersebut harus berada di akun yang sama dengan API. Variabel tahap tidak mendukung fungsi Lambda lintas akun.

AWS kredensi integrasi

Anda dapat menggunakan variabel tahap sebagai bagian dari ARN kredensi AWS pengguna atau peran, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Terapkan WebSocket APIs di API Gateway

Setelah membuat WebSocket API Anda, Anda harus menerapkannya agar tersedia bagi pengguna Anda untuk dipanggil.

[Untuk menerapkan API, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung.](#) Setiap tahap adalah snapshot dari API dan tersedia untuk aplikasi klien untuk dipanggil.

Important

Setiap kali Anda memperbarui API, Anda harus menerapkannya kembali. Perubahan pada apa pun selain pengaturan tahap memerlukan pemindahan, termasuk modifikasi pada sumber daya berikut:

- Rute
- Integrasi
- Pengotorisasi

Secara default, Anda dibatasi hingga 10 tahap untuk setiap API. Kami menyarankan Anda menggunakan kembali tahapan untuk penerapan Anda.

Untuk memanggil WebSocket API yang diterapkan, klien mengirim pesan ke URL API. URL ditentukan oleh nama host dan nama panggung API.

Note

API Gateway akan mendukung payload hingga 128 KB dengan ukuran frame maksimal 32 KB. Jika pesan melebihi 32 KB, itu harus dibagi menjadi beberapa frame, masing-masing 32 KB atau lebih kecil.

Menggunakan nama domain default API, URL (misalnya WebSocket API dalam tahap tertentu (`{stageName}`)) adalah dalam format berikut:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Untuk membuat URL WebSocket API lebih user-friendly, Anda dapat membuat nama domain kustom (misalnya, `api.example.com`) untuk mengganti nama host default API. Proses konfigurasi sama dengan REST APIs. Untuk informasi selengkapnya, lihat [the section called “Nama domain kustom”](#).

Tahapan memungkinkan kontrol versi yang kuat dari API Anda. Misalnya, Anda dapat menerapkan API ke test stage dan prod stage, dan menggunakan test stage sebagai test build dan menggunakan prod stage sebagai build stabil. Setelah pembaruan lulus tes, Anda dapat mempromosikan test panggung ke prod panggung. Promosi dapat dilakukan dengan memindahkan API ke panggung. prod Untuk detail lebih lanjut tentang tahapan, lihat [the section called “Siapkan panggung”](#).

Topik

- [Membuat penerapan WebSocket API menggunakan AWS CLI](#)
- [Membuat penerapan WebSocket API menggunakan konsol API Gateway](#)

Membuat penerapan WebSocket API menggunakan AWS CLI

Perintah [create-deployment](#) berikut membuat deployment:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccdde
```

Outputnya akan terlihat seperti berikut:

```
{
  "DeploymentId": "fedcba",
  "DeploymentStatus": "DEPLOYED",
  "CreateDate": "2018-11-15T06:49:09Z"
}
```

API yang diterapkan tidak dapat dipanggil sampai Anda mengaitkan penerapan dengan tahapan. Anda dapat membuat tahap baru atau menggunakan kembali tahap yang telah Anda buat sebelumnya.

Perintah [create-stage](#) berikut membuat tahap baru dan mengaitkannya dengan penerapan:


```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccdde --deployment-id fedcba --stage-name test
```

Outputnya terlihat seperti berikut:

```
{
  "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": {}
}
```

Anda juga dapat menggunakan kembali tahap yang ada dengan memperbarui deploymentId properti stage dengan ID penerapan () *deployment-id* yang baru dibuat. Perintah [update-stage](#) berikut memperbarui ID penyebaran tahap:

```
aws apigatewayv2 update-stage --region region \
  --api-id api-id \
  --stage-name stage-name \
  --deployment-id deployment-id
```

Membuat penerapan WebSocket API menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk membuat penerapan WebSocket API:

1. Masuk ke konsol API Gateway dan pilih API.
2. Pilih Deploy API.
3. Pilih tahap yang diinginkan dari daftar dropdown atau masukkan nama tahap baru.

Kebijakan keamanan untuk WebSocket APIs di API Gateway

API Gateway memberlakukan kebijakan keamanan TLS_1_2 untuk semua titik akhir WebSocket API.

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh Amazon API Gateway. Protokol TLS mengatasi masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server. Ketika klien Anda membuat jabat tangan TLS ke API Anda melalui domain kustom, kebijakan keamanan memberlakukan versi TLS dan pilihan cipher suite yang dapat dipilih klien Anda untuk digunakan. Kebijakan keamanan ini menerima lalu lintas TLS 1.2 dan TLS 1.3 dan menolak lalu lintas TLS 1.0.

Protokol dan cipher TLS yang didukung untuk WebSocket APIs

Tabel berikut menjelaskan protokol TLS yang didukung untuk WebSocket APIs

Protokol TLS	Kebijakan keamanan TLS_1_2
TLSv1.3	Ya
TLSv1.2	Ya

Tabel berikut menjelaskan cipher TLS yang tersedia untuk kebijakan keamanan TLS 1_2 untuk WebSocket APIs

Cipher TLS	Kebijakan keamanan TLS_1_2
TLS_AES_128_GCM_SHA256	Ya
TLS_AES_256_GCM_SHA384	Ya
TLS_ _ _ CHACHA20 POLY1305 SHA256	Ya
ECDHE-ECDSA- -GCM- AES128 SHA256	Ya
ECDHE-RSA- -GCM- AES128 SHA256	Ya
ECDHE-ECDSA- - AES128 SHA256	Ya
ECDHE-RSA- - AES128 SHA256	Ya
ECDHE-ECDSA- -GCM- AES256 SHA384	Ya

Cipher TLS	Kebijakan keamanan TLS_1_2
ECDHE-RSA- -GCM- AES256 SHA384	Ya
ECDHE-ECDSA- - AES256 SHA384	Ya
ECDHE-RSA- - AES256 SHA384	Ya
AES128-GCM- SHA256	Ya
AES128-SHA256	Ya
AES256-GCM- SHA384	Ya
AES256-SHA256	Ya

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246, menggunakan nama yang berbeda untuk cipher yang sama. Untuk daftar nama sandi, lihat [the section called “Nama sandi OpenSSL dan RFC”](#)

Informasi tentang REST APIs dan HTTP APIs

Untuk informasi lebih lanjut tentang REST APIs dan HTTP APIs, lihat [the section called “Pilih kebijakan keamanan”](#) dan [the section called “Kebijakan keamanan untuk HTTP APIs di API Gateway”](#).

Nama domain khusus untuk WebSocket APIs di API Gateway

Nama domain khusus lebih sederhana dan lebih intuitif URLs yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API Anda, Anda (dan pelanggan Anda) dapat memanggil API menggunakan URL dasar default dari format berikut:

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

dimana *api-id* dihasilkan oleh API Gateway, *region* adalah AWS Region, dan *stage* ditentukan oleh Anda saat menerapkan API.

Bagian nama host dari URL, *api-id*.execute-api.*region*.amazonaws.com mengacu pada titik akhir API. Nama endpoint API default dibuat secara acak, sulit diingat, dan tidak ramah pengguna.

Dengan nama domain khusus, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya, *myservice*) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

Pertimbangan

Pertimbangan berikut dapat memengaruhi penggunaan nama domain kustom Anda.

- Jika Anda memetakan nama domain kustom ke WebSocket API, Anda tidak dapat memetakannya ke REST API atau HTTP API.
- Hanya nama domain kustom Regional yang didukung.
- Untuk versi TLS minimum, hanya TLS 1.2 yang didukung.
- Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS Anda untuk dipetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain khusus tidak dapat mencapai API Gateway.
- Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default dengan menggunakan sertifikat wildcard. Untuk informasi selengkapnya, lihat [the section called “Nama domain kustom wildcard”](#).

Prasyarat

Berikut ini adalah prasyarat untuk nama domain kustom.

Daftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk mengatur nama domain khusus untuk Anda APIs. Anda dapat mendaftarkan nama domain internet Anda menggunakan [Amazon Route 53](#) atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom Anda dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zone apex”) dari domain internet terdaftar.

Nama domain Anda harus mengikuti spesifikasi [RFC 1035](#) dan dapat memiliki maksimum 63 oktet per label dan total 255 oktet.

Sertifikat untuk nama domain kustom

Sebelum menyiapkan nama domain khusus untuk API, Anda harus memiliki sertifikat SSL/TLS yang siap di ACM. Jika ACM tidak tersedia di AWS Wilayah tempat Anda membuat nama domain kustom, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor badan SSL/TLS certificate, you must provide the PEM-formatted SSL/TLS sertifikat, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom.

Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir mengekspos detail sertifikat sensitif apa pun, seperti kunci pribadi. Untuk menggunakan sertifikat AWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

Jika aplikasi Anda menggunakan pinning sertifikat, kadang-kadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelah AWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat [Masalah penyematan sertifikat](#) di Panduan AWS Certificate Manager Pengguna.

Nama domain kustom wildcard

Dengan nama domain khusus wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota [default](#). Misalnya, Anda bisa memberi setiap pelanggan Anda nama domain mereka sendiri `customername.api.example.com`.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (*) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard `*.example.com` menghasilkan subdomain seperti `example.com`, dan `b.example.com`. `c.example.com`, yang semuanya merutekan ke domain yang sama.

Nama domain kustom wildcard mendukung konfigurasi yang berbeda dari nama domain kustom standar API Gateway. Misalnya, dalam satu AWS akun, Anda dapat mengkonfigurasi `*.example.com` dan `a.example.com` berperilaku berbeda.

Anda dapat menggunakan variabel `$context.domainName` dan `$context.domainPrefix` konteks untuk menentukan nama domain yang digunakan klien untuk memanggil API Anda. Untuk mempelajari lebih lanjut tentang variabel konteks, lihat [Variabel untuk transformasi data untuk API Gateway](#).

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

Note

Anda tidak dapat membuat nama domain khusus wildcard jika AWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuat `.example.com`, maka akun B tidak dapat membuat nama `*.example.com` domain khusus wildcard.

Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi [Pusat AWS Dukungan](#) untuk meminta pengecualian.

Langkah selanjutnya untuk nama domain kustom

Untuk menyiapkan nama domain kustom untuk API HTTP, Anda menggunakan dokumentasi dari bagian REST API dari Panduan Pengembang API Gateway.

Pertama, tentukan sertifikat untuk nama domain kustom Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan sertifikat AWS Certificate Manager”](#). Selanjutnya, Anda membuat nama domain kustom Regional. Lihat informasi yang lebih lengkap di [the section called “Siapkan nama domain kustom Regional”](#).

Memetakan tahapan API ke nama domain khusus untuk WebSocket APIs

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain kustom. Setelah Anda membuat nama domain dan mengonfigurasi catatan DNS, Anda menggunakan pemetaan API untuk mengirim lalu lintas ke Anda APIs melalui nama domain kustom Anda.

Pemetaan API menentukan API, tahap, dan jalur opsional yang akan digunakan untuk pemetaan. Misalnya, Anda dapat memetakan `production` tahap API ke `wss://api.example.com/orders`.

Sebelum membuat pemetaan API, Anda harus memiliki API, panggung, dan nama domain khusus. Untuk mempelajari selengkapnya tentang membuat nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Pembatasan

- Dalam pemetaan API, nama domain khusus dan dipetakan APIs harus berada di akun yang sama AWS.

- Pemetaan API harus hanya berisi huruf, angka, dan karakter berikut: `$-_.+!*'()`
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda tidak dapat memetakan WebSocket APIs ke nama domain kustom yang sama dengan HTTP API atau REST API.
- Jika Anda membuat pemetaan API dengan beberapa level, API Gateway mengonversi semua nama header menjadi huruf kecil.

Buat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Untuk informasi tentang membuat nama domain kustom, lihat [the section called “Siapkan nama domain kustom Regional”](#).

Konsol Manajemen AWS

Untuk membuat pemetaan API

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih Nama domain khusus.
3. Pilih nama domain khusus yang telah Anda buat.
4. Pilih pemetaan API.
5. Pilih Konfigurasi pemetaan API.
6. Pilih Tambahkan pemetaan baru.
7. Masukkan API, Stage, dan secara opsional Path.
8. Pilih Simpan.

AWS CLI

[create-api-mapping](#) Perintah berikut membuat pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan `api.example.com/v1` ke API dan tahap yang ditentukan.

```
aws apigatewayv2 create-api-mapping \  
  --domain-name api.example.com \  
  --api-mapping-key v1 \  
  --api-id a1b2c3d4 \  

```

```
--stage test
```

CloudFormation

CloudFormation Contoh berikut membuat pemetaan API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Jenis alamat IP untuk nama domain khusus untuk WebSocket APIs

Saat Anda membuat nama domain khusus, Anda menentukan jenis alamat IP yang dapat memanggil domain Anda. Anda dapat memilih IPv4 untuk menyelesaikan IPv4 alamat untuk memanggil domain Anda, atau Anda dapat memilih dualstack untuk mengizinkan keduanya IPv4 dan IPv6 alamat memanggil domain Anda. Kami menyarankan Anda mengatur jenis alamat IP ke dualstack untuk mengurangi kelelahan ruang IP atau untuk postur keamanan Anda. [Untuk informasi selengkapnya tentang manfaat dari jenis alamat IP dualstack, lihat IPv6 di. AWS](#)

Pertimbangan untuk jenis alamat IP

Pertimbangan berikut dapat memengaruhi penggunaan jenis alamat IP Anda.

- Jenis alamat IP default untuk nama domain kustom API Gateway adalah IPv4.
- Nama domain kustom Anda tidak perlu memiliki jenis alamat IP yang sama untuk semua yang APIs dipetakan ke sana. Jika Anda menonaktifkan titik akhir API default, hal ini dapat memengaruhi cara penelepon dapat menjalankan API Anda.

Mengubah jenis alamat IP dari nama domain kustom

Anda dapat mengubah jenis alamat IP dengan memperbarui konfigurasi titik akhir nama domain. Anda dapat memperbarui konfigurasi titik akhir dengan menggunakan Konsol Manajemen AWS,, AWS CLI CloudFormation, atau AWS SDK.

Konsol Manajemen AWS

Untuk mengubah jenis alamat IP dari nama domain kustom

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih nama domain kustom publik.
3. Pilih konfigurasi Endpoint.
4. Untuk jenis alamat IP, pilih salah satu IPv4 atau Dualstack.
5. Pilih Simpan.

AWS CLI

[update-domain-name](#) Perintah berikut memperbarui API untuk memiliki tipe alamat IP dualstack:

```
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
```

Outputnya akan terlihat seperti berikut:

```
{  
  "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": {}  
}
```

Nonaktifkan titik akhir default untuk WebSocket APIs

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus, nonaktifkan `execute-api` titik akhir default. Saat Anda menonaktifkan titik akhir default, itu memengaruhi semua tahapan API.

Prosedur berikut menunjukkan cara menonaktifkan titik akhir default untuk WebSocket API.

Konsol Manajemen AWS

1. Masuk ke konsol API Gateway di <https://console.aws.amazon.com/apigateway>.
2. Pilih WebSocket API.
3. Pilih setelan API.
4. Pada detail API, pilih Edit.
5. Untuk titik akhir Default, pilih Tidak Aktif.
6. Pilih Simpan perubahan.
7. Pada panel navigasi utama, pilih Rute.
8. Pilih Deploy, lalu terapkan ulang API Anda atau buat tahap baru agar perubahan diterapkan.

AWS CLI

Perintah [update-api](#) berikut menonaktifkan titik akhir default untuk API: WebSocket

```
aws apigatewayv2 update-api \  
  --api-id abcdef123 \  
  --disable-execute-api-endpoint
```

Setelah menonaktifkan titik akhir default, Anda harus menerapkan API agar perubahan diterapkan.

AWS CLI Perintah berikut membuat penyebaran.

```
aws apigatewayv2 create-deployment \  
  --api-id abcdef123 \  
  --stage-name dev
```

Lindungi Anda WebSocket APIs di API Gateway

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti pengguna jahat atau lonjakan lalu lintas. Anda dapat melindungi API menggunakan strategi seperti membuat sertifikat SSL atau menetapkan target pelambatan. Untuk informasi selengkapnya tentang membuat sertifikat SSL, lihat [the section called “Sertifikat Klien”](#). Sisa bagian ini mencakup pengaturan target pelambatan.

Anda dapat mengonfigurasi pembatasan APIs untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Throttle diterapkan atas dasar upaya terbaik dan harus dianggap sebagai target daripada plafon permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritme token bucket, tempat token diperhitungkan untuk permintaan. Secara khusus, API Gateway memeriksa tarif dan ledakan pengiriman permintaan terhadap semua yang ada APIs di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batas dikuasai dalam beberapa kasus.

Jika pengiriman permintaan melebihi tingkat permintaan kondisi tunak dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima tanggapan 429 Too Many Requests kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan atau rute API untuk meningkatkan kinerja keseluruhan APIs di semua akun Anda.

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan kondisi tunak per detik (RPS) APIs di semua AWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua APIs dalam AWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target pengiriman permintaan bersamaan yang akan dipenuhi API Gateway sebelum mengembalikan respons 429 Too Many Requests kesalahan. Untuk informasi lebih lanjut tentang pembatasan kuota, lihat. [Kuota dan catatan penting](#)

Batas per akun diterapkan ke semua akun APIs di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan batas waktu APIs yang lebih pendek dan muatan yang lebih kecil. [Untuk meminta peningkatan batas pembatasan](#)

[tingkat akun per Wilayah, hubungi Pusat Dukungan.AWS](#) Untuk informasi selengkapnya, lihat [Kuota dan catatan penting](#). Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.

Pelambatan tingkat rute

Anda dapat menyetel pembatasan tingkat rute untuk mengganti batas pembatasan permintaan tingkat akun untuk tahap tertentu atau untuk rute individual di API Anda. Batas pembatasan rute default tidak dapat melebihi batas tingkat akun.

Anda dapat mengonfigurasi pelambatan tingkat rute dengan menggunakan file. AWS CLI Perintah [update-stage](#) berikut mengonfigurasi pelambatan kustom untuk tahap dan rute API yang ditentukan:

```
aws apigatewayv2 update-stage \  
  --api-id a1b2c3d4 \  
  --stage-name dev \  
  --route-settings '{"messages":  
{ "ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000 } }'
```

Pantau WebSocket APIs di API Gateway

Anda dapat menggunakan CloudWatch metrik dan CloudWatch Log untuk memantau WebSocket APIs. Dengan menggabungkan log dan metrik, Anda dapat mencatat kesalahan dan memantau kinerja API Anda.

Note

API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:

- 413 Kesalahan Permintaan Entitas Terlalu Besar
- Berlebihan 429 Terlalu Banyak Kesalahan Permintaan
- 400 error seri dari permintaan yang dikirim ke domain kustom yang tidak memiliki pemetaan API
- 500 kesalahan seri yang disebabkan oleh kegagalan internal

Topik

- [Pantau eksekusi WebSocket API dengan CloudWatch metrik](#)
- [Konfigurasi logging untuk WebSocket APIs di API Gateway](#)

Pantau eksekusi WebSocket API dengan CloudWatch metrik

Anda dapat menggunakan CloudWatch metrik [Amazon](#) untuk memantau WebSocket APIs. Konfigurasinya mirip dengan yang digunakan untuk REST APIs. Untuk informasi selengkapnya, lihat [Pantau eksekusi REST API dengan CloudWatch metrik Amazon](#).

Metrik berikut didukung untuk WebSocket APIs:

Metrik	Deskripsi
ConnectCount	Jumlah pesan yang dikirim ke integrasi \$connect rute.
MessageCount	Jumlah pesan yang dikirim ke WebSocket API, baik dari atau ke klien.
IntegrationError	Jumlah permintaan yang mengembalikan respons 4XX/5XX dari integrasi.
ClientError	Jumlah permintaan yang memiliki respons 4XX yang dikembalikan oleh API Gateway sebelum integrasi dipanggil.
ExecutionError	Kesalahan yang terjadi saat memanggil integrasi.
IntegrationLatency	Perbedaan waktu antara API Gateway mengirimk an permintaan ke integrasi dan API Gateway menerima respons dari integrasi. Ditekan untuk panggilan balik dan integrasi tiruan.

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

Dimensi	Deskripsi
Apild	Memfilter metrik API Gateway untuk API dengan ID API yang ditentukan.
Apild, Panggung	Memfilter metrik API Gateway untuk tahap API dengan ID API dan ID tahap yang ditentukan.
Apild, Metode, Sumber Daya, Panggung	Memfilter metrik API Gateway untuk metode API dengan ID API, ID tahap, jalur sumber daya, dan ID rute yang ditentukan. API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Anda dapat melakukan ini dengan memanggil UpdateStage aksi API Gateway V2 REST API untuk memperbarui <code>detailedMetricsEnabled</code> properti ke <code>true</code>. Atau, Anda dapat memanggil AWS CLI perintah update-stage untuk memperbarui properti ke <code>DetailedMetricsEnabled true</code>. Mengaktifkan metrik tersebut akan dikenakan biaya tambahan ke akun Anda. Untuk informasi

Dimensi	Deskripsi
	harga, lihat CloudWatchHarga Amazon .

Konfigurasikan logging untuk WebSocket APIs di API Gateway

Anda dapat mengaktifkan logging untuk menulis log ke CloudWatch Log. Ada dua jenis log masuk API CloudWatch: logging eksekusi dan logging akses. Dalam pencatatan eksekusi, API Gateway mengelola CloudWatch Log. Prosesnya mencakup pembuatan grup log dan aliran log, dan pelaporan ke aliran log permintaan dan tanggapan pemanggil apa pun.

Untuk meningkatkan postur keamanan Anda, kami sarankan Anda menggunakan pencatatan eksekusi di INFO tingkat ERROR atau. Anda mungkin perlu melakukan ini untuk mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya, lihat [kontrol Amazon API Gateway](#) di Panduan AWS Security Hub Pengguna.

Dalam pencatatan akses, Anda, sebagai pengembang API, ingin mencatat siapa yang telah mengakses API Anda dan bagaimana pemanggil mengakses API. Anda dapat membuat grup log Anda sendiri atau memilih grup log yang sudah ada yang dapat dikelola oleh API Gateway. Untuk menentukan rincian akses, Anda memilih `$context` variabel (dinyatakan dalam format pilihan Anda) dan memilih grup log sebagai tujuan.

Untuk petunjuk tentang cara mengatur CloudWatch logging, lihat [the section called “Siapkan pencatatan CloudWatch API menggunakan konsol API Gateway”](#).

Saat Anda menentukan Format Log, Anda dapat memilih variabel konteks mana yang akan dicatat. Variabel berikut didukung.

Parameter	Deskripsi
<code>\$context.apiId</code>	API Gateway identifier ditetapkan ke API Anda.
<code>\$context.authorize.error</code>	Pesan kesalahan otorisasi.
<code>\$context.authorize.latency</code>	Latensi otorisasi di ms.
<code>\$context.authorize.status</code>	Kode status dikembalikan dari upaya otorisasi.

Parameter	Deskripsi
<code>\$context.authorizer.error</code>	Pesan kesalahan dikembalikan dari otorisasi.
<code>\$context.authorizer.integrationLatency</code>	Latensi otorisasi Lambda di ms.
<code>\$context.authorizer.integrationStatus</code>	Kode status dikembalikan dari otorisasi Lambda.
<code>\$context.authorizer.latency</code>	Latensi otorisasi di ms.
<code>\$context.authorizer.requestId</code>	ID permintaan AWS titik akhir.
<code>\$context.authorizer.status</code>	Kode status dikembalikan dari otorisasi.
<code>\$context.authorizer.principalId</code>	Identifikasi pengguna utama yang dikaitkan dengan token yang dikirim oleh klien dan dikembalikan dari fungsi Lambda otorisasi API Gateway Lambda. (Authorizer Lambda sebelumnya dikenal sebagai otorisasi khusus.)

Parameter	Deskripsi
<code>\$context.authorizer.</code> <i>property</i>	Nilai stringifikasi dari pasangan nilai kunci yang ditentukan dari context peta dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan context peta berikut: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> memanggil <code>\$context.authorizer.key</code> mengembalikan "value" string, memanggil <code>\$context.authorizer.numKey</code> mengembalikan "1" string, dan memanggil <code>\$context.authorizer.boolKey</code> mengembalikan "true" string.
<code>\$context.authenticate.error</code>	Pesan kesalahan dikembalikan dari upaya otentikasi.
<code>\$context.authenticate.latency</code>	Latensi otentikasi di ms.
<code>\$context.authenticate.status</code>	Kode status dikembalikan dari upaya otentikasi.
<code>\$context.connectedAt</code>	Waktu koneksi yang diformat Epoch .
<code>\$context.connectionId</code>	ID unik untuk koneksi yang dapat digunakan untuk membuat callback ke klien.
<code>\$context.domainName</code>	Sebuah nama domain untuk WebSocket API. Ini dapat digunakan untuk membuat panggilan balik ke klien (bukan nilai hardcoded).

Parameter	Deskripsi
<code>\$context.error.message</code>	String yang berisi pesan kesalahan API Gateway.
<code>\$context.error.messageString</code>	Nilai yang dikutip dari <code>\$context.error.message</code> , yaitu <code>"\$context.error.message"</code> .
<code>\$context.error.responseType</code>	Jenis respons kesalahan.
<code>\$context.error.validationErrorString</code>	Sebuah string yang berisi pesan kesalahan validasi rinci.
<code>\$context.eventType</code>	Jenis acara: <code>CONNECT</code> , <code>MESSAGE</code> , atau <code>DISCONNECT</code> .
<code>\$context.extendedRequestId</code>	Setara dengan <code>\$context.requestId</code> .
<code>\$context.identity.accountId</code>	ID AWS akun yang terkait dengan permintaan.
<code>\$context.identity.apiKey</code>	Kunci pemilik API yang terkait dengan permintaan API berkemampuan kunci.
<code>\$context.identity.apiKeyId</code>	ID kunci API yang terkait dengan permintaan API berkemampuan kunci
<code>\$context.identity.caller</code>	Pengenal utama penelepon yang menandatangani permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.

Parameter	Deskripsi
<code>\$context.identity.cognitoAuthenticationProvider</code>	Daftar dipisahkan koma dari semua penyedia otentikasi Amazon Cognito yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></code> Untuk informasi tentang penyedia autentikasi Amazon Cognito yang tersedia, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito. Nilai yang mungkin termasuk <code>authenticated</code> untuk identitas yang diautentikasi dan <code>unauthenticated</code> untuk identitas yang tidak diautentikasi.
<code>\$context.identity.cognitoIdentityId</code>	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensial Amazon Cognito.

Parameter	Deskripsi
<code>\$context.identity.principalOrgId</code>	ID AWS organisasi . Didukung untuk rute yang menggunakan otorisasi IAM.
<code>\$context.identity.sourceIp</code>	Alamat IP sumber koneksi TCP membuat permintaan ke API Gateway.
<code>\$context.identity.user</code>	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk rute yang menggunakan otorisasi IAM.
<code>\$context.identity.userAgent</code>	Agen pengguna pemanggil API.
<code>\$context.identity.userArn</code>	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi.
<code>\$context.integration.error</code>	Pesan kesalahan dikembalikan dari integrasi.
<code>\$context.integration.integrationStatus</code>	Untuk integrasi proxy Lambda, kode status dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
<code>\$context.integration.latency</code>	Latensi integrasi dalam ms. Setara dengan <code>\$context.integrationLatency</code> .
<code>\$context.integration.requestId</code>	ID permintaan AWS titik akhir. Setara dengan <code>\$context.awsEndpointRequestId</code> .
<code>\$context.integration.status</code>	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda. Setara dengan <code>\$context.integrationStatus</code> .

Parameter	Deskripsi
<code>\$context.integrationLatency</code>	Latensi integrasi dalam ms, hanya tersedia untuk pencatatan akses.
<code>\$context.messageId</code>	ID sisi server unik untuk pesan. Hanya tersedia ketika <code>\$context.eventType</code> ada <code>MESSAGE</code> .
<code>\$context.requestId</code>	Sama seperti <code>\$context.extendedRequestId</code> .
<code>\$context.requestTime</code>	Waktu permintaan yang diformat CLF (). dd/MMM/yyyy:HH:mm:ss +-hhmm
<code>\$context.requestTimeEpoch</code>	Waktu permintaan yang diformat Epoch , dalam milidetik.
<code>\$context.routeKey</code>	Kunci rute yang dipilih.
<code>\$context.stage</code>	Tahap penerapan panggilan API (misalnya, beta atau prod).
<code>\$context.status</code>	Status respon.
<code>\$context.waf.error</code>	Pesan kesalahan dikembalikan dari AWS WAF.
<code>\$context.waf.latency</code>	AWS WAF Latensi dalam ms.
<code>\$context.waf.status</code>	Kode status dikembalikan dari AWS WAF.

Contoh beberapa format log akses yang umum digunakan ditampilkan di konsol API Gateway dan dicantumkan sebagai berikut.

- CLF([Format Log Umum](#)):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
$context.connectionId" \
```

```
$context.status $context.requestId
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- 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"
}
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- 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>
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- CSV(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.identity.caller, \  
$context.identity.user,$context.requestTime,$context.eventType, \  
$context.routeKey,$context.connectionId,$context.status, \  
$context.requestId
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

Referensi API Gateway Amazon Resource Name (ARN)

Tabel berikut mencantumkan Amazon Resource Names (ARNs) untuk resource API Gateway. Untuk mempelajari lebih lanjut tentang penggunaan ARNs dalam AWS Identity and Access Management kebijakan, lihat [Cara kerja Amazon API Gateway dengan IAM](#) dan [Kontrol akses ke REST API dengan izin IAM](#).

Sumber daya API dan WebSocket API HTTP

Sumber daya	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
Pengotorisasi	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /authoriz ers/ <i>id</i>
Pengotorisasi	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
Deployment	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /deployme nts/ <i>id</i>

Sumber daya	ARN
Deployment	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>
Integrasi	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /integrations/ <i>integration-id</i>
Integrasi	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
Model	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/apis/ <i>api-id</i> /models/ <i>id</i>
Model	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>

Sumber daya	ARN
Route	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>
Tahapan	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

Sumber daya REST API

Sumber daya	ARN
Akun	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
Pengotorisasi	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / authorizers/ <i>id</i>
Pengotorisasi	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
Deployment	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / deployments/ <i>id</i>
Deployment	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / deployments

Sumber daya	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
Integrasi	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>

Sumber daya	ARN
Metode	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>
Model	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / models/ <i>model-name</i>
Model	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
Sumber daya	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources/ <i>id</i>
Sumber daya	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / resources

Sumber daya	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>
Tahapan	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/restapis/ <i>api-id</i> / stages
Tag	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/tags/ <i>url-encoded-resource-arn</i>
Templat	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

Sumber daya	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

Sumber nama domain

Sumber daya	ARN
ApiMapping	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-na</i> <i>me</i> /apimappings/ <i>id</i>
ApiMappings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-na</i> <i>me</i> /apimappings
BasePathMapping	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-na</i> <i>me</i> /basepathmappings/ <i>basepath</i>
BasePathMappings	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-na</i> <i>me</i> /basepathmappings
DomainName	arn: <i>partition</i> :apigatew ay: <i>region</i> ::/domainnames/ <i>domain-na</i> <i>me</i>
DomainNameAccessAssociation	arn: <i>partition</i> :apigatew ay: <i>region</i> : <i>account-id</i> :/domainn ameaccessassociations/domai nname/ <i>domain-name</i> /vpcesour ce/ <i>vpce-source-id</i>

Sumber daya	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, dan REST APIs)

Sumber daya	ARN
WebSocket Titik akhir API	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> /stage/route-key
HTTP API dan titik akhir REST API*	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> /stage/http-method /resource-path
Pengotorisasi Lambda**	arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> /authorizers/ <i>authorizer-id</i>

* ARN untuk titik akhir \$default rute untuk HTTP adalah. APIs arn:*partition*:execute-api:*region*:*account-id*:*api-id*/*/\$default

** ARN ini hanya berlaku saat menyetel SourceArn kondisi dalam [kebijakan sumber daya untuk fungsi otorisasi](#) Lambda. Sebagai contoh, lihat [the section called “Buat Authorizer Lambda”](#).

Ekstensi OpenAPI untuk API Gateway

Ekstensi API Gateway mendukung otorisasi AWS khusus dan integrasi API khusus Gateway API untuk REST dan HTTP. APIs APIs Di bagian ini, kami menjelaskan ekstensi API Gateway ke spesifikasi OpenAPI.

Tip

Untuk memahami bagaimana ekstensi API Gateway digunakan dalam aplikasi, Anda dapat menggunakan konsol API Gateway untuk membuat REST API atau HTTP API dan mengekspornya ke file definisi OpenAPI. Untuk informasi selengkapnya tentang cara mengekspor API, lihat [Ekspor REST API dari API Gateway](#) dan [Ekspor HTTP APIs dari API Gateway](#).

Topik

- [x-amazon-apigateway-anyobjek -metode](#)
- [x-amazon-apigateway-cors objek](#)
- [x-amazon-apigateway-apiproperti -key-source](#)
- [x-amazon-apigateway-auth objek](#)
- [x-amazon-apigateway-authorizer objek](#)
- [x-amazon-apigateway-authtype properti](#)
- [x-amazon-apigateway-binaryproperti -media-tipe](#)
- [x-amazon-apigateway-documentation objek](#)
- [x-amazon-apigateway-endpoint-modus akses](#)
- [x-amazon-apigateway-endpoint-konfigurasi objek](#)
- [x-amazon-apigateway-gateway-tanggapan objek](#)
- [x-amazon-apigateway-gateway-Responses.gatewayResponse objek](#)
- [x-amazon-apigateway-gateway-Responses.responseParameters objek](#)
- [x-amazon-apigateway-gateway-Responses.responseTemplates objek](#)
- [x-amazon-apigateway-importexport-versi](#)
- [x-amazon-apigateway-integration objek](#)

- [x-amazon-apigateway-integrations objek](#)
- [x-amazon-apigateway-integrationObjek. RequestTemplates](#)
- [x-amazon-apigateway-integrationObjek. RequestParameters](#)
- [x-amazon-apigateway-integration.response objek](#)
- [x-amazon-apigateway-integration.response objek](#)
- [x-amazon-apigateway-integrationObjek. ResponseTemplates](#)
- [x-amazon-apigateway-integration.ResponseParameters objek](#)
- [x-amazon-apigateway-integrationObjek .tlsConfig](#)
- [x-amazon-apigateway-minimum-ukuran kompresi](#)
- [x-amazon-apigateway-policy](#)
- [x-amazon-apigateway-requestproperty -validator](#)
- [x-amazon-apigateway-request-validator objek](#)
- [x-amazon-apigateway-request-Validators.requestValidator objek](#)
- [x-amazon-apigateway-security-kebijakan](#)
- [x-amazon-apigateway-tag-nilai properti](#)

x-amazon-apigateway-anyobjek -metode

Menentukan Objek [Operasi OpenAPI](#) untuk ANY metode catch-all API Gateway dalam Objek Item Jalur [OpenAPI](#). Objek ini dapat eksis bersama objek Operasi lainnya dan akan menangkap metode HTTP apa pun yang tidak dideklarasikan secara eksplisit.

Tabel berikut mencantumkan properti yang diperluas oleh API Gateway. Untuk properti Operasi OpenAPI lainnya, lihat spesifikasi OpenAPI.

Nama properti	Tipe	Deskripsi
<code>isDefaultRoute</code>	Boolean	Menentukan apakah rute adalah \$default rute. Hanya didukung untuk HTTP APIs. Untuk mempelajari selengkapnya, lihat Buat rute untuk HTTP APIs di API Gateway .

Nama properti	Tipe	Deskripsi
x-amazon-apigateway-integration	x-amazon-apigateway-integration objek	Menentukan integrasi metode dengan backend. Ini adalah properti yang diperluas dari objek Operasi OpenAPI . Integrasi dapat berupa tipeAWS,AWS_PROXY ,HTTP,HTTP_PROXY , atauMOCK.

x-amazon-apigateway-any-contoh metode

Contoh berikut mengintegrasikan ANY metode pada sumber daya proxy,{proxy+}, dengan fungsi Lambda,. TestSimpleProxy

```
"/{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"
    }
  }
}
```

Contoh berikut membuat \$default rute untuk HTTP API yang terintegrasi dengan fungsi Lambda,. HelloWorld

```
"/$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 objek

Menentukan konfigurasi cross-origin resource sharing (CORS) untuk HTTP API. Ekstensi ini berlaku untuk struktur OpenAPI tingkat root. Untuk mempelajari selengkapnya, lihat [Konfigurasi CORS untuk HTTP APIs di API Gateway](#).

Nama properti	Tipe	Deskripsi
allowOrigins	Array	Menentukan asal-usul diizinkan.
allowCredentials	Boolean	Menentukan apakah kredensial termasuk dalam permintaan CORS.
exposeHeaders	Array	Menentukan header yang diekspos.
maxAge	Integer	Menentukan jumlah detik bahwa browser harus cache hasil permintaan preflight.
allowMethods	Array	Menentukan metode HTTP diperbolehkan.

Nama properti	Tipe	Deskripsi
allowHeaders	Array	Menentukan header diperbolehkan.

x-amazon-apigateway-cors contoh

Berikut ini adalah contoh konfigurasi CORS untuk 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-apiproperty-key-source

Tentukan sumber untuk menerima kunci API untuk membatasi metode API yang memerlukan kunci. Properti API-level ini adalah tipe. String Untuk informasi selengkapnya tentang mengonfigurasi metode agar memerlukan kunci API, lihat [the section called “Konfigurasi metode untuk menggunakan kunci API dengan definisi OpenAPI”](#).

Tentukan sumber kunci API untuk permintaan. Nilai yang valid adalah:

- **HEADER** untuk menerima kunci API dari X-API-Key header permintaan.
- **AUTHORIZER** untuk menerima kunci API dari `UsageIdentifierKey` dari pengotorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus).

x-amazon-apigateway-api-contoh sumber kunci

Contoh berikut menetapkan X-API-Key header sebagai sumber kunci API.

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 objek

Mendefinisikan jenis otorisasi yang akan diterapkan untuk otorisasi pemanggilan metode di API Gateway.

Nama properti	Tipe	Deskripsi
type	string	Menentukan jenis otorisasi . Tentukan "NONE" untuk akses terbuka. Tentukan "AWS_IAM" untuk menggunakan izin IAM. Nilai tidak peka huruf besar/kecil.

x-amazon-apigateway-auth contoh

Contoh berikut menetapkan jenis otorisasi untuk metode API.

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 objek

Mendefinisikan otorisasi Lambda, kumpulan pengguna Amazon Cognito, atau otorisasi JWT yang akan diterapkan untuk otorisasi pemanggilan metode di API Gateway. [Ekstensi ini berlaku untuk definisi keamanan di OpenAPI 2 dan skema keamanan di OpenAPI 3.](#)

Nama properti	Tipe	Deskripsi
type	string	Jenis otorisasi. Ini adalah properti yang diperlukan. Untuk REST APIs, tentukan token otorisasi dengan identitas pemanggil yang disematkan dalam token otorisasi. Tentukan request otorisasi dengan identitas pemanggil yang terkandung dalam parameter permintaan. Tentukan <code>cognito_user_pools</code> otorisasi yang menggunakan kumpulan pengguna Amazon Cognito untuk mengontrol akses ke API Anda. Untuk HTTP APIs, tentukan request otorisasi Lambda dengan identitas pemanggil yang terkandung dalam parameter permintaan. Tentukan <code>jwt</code> untuk otorisasi JWT.

Nama properti	Tipe	Deskripsi
<code>authorizerUri</code>	<code>string</code>	Uniform Resource Identifier (URI) dari fungsi Lambda authorizer. Sintaksnya adalah sebagai berikut: <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>
<code>authorizerCredentials</code>	<code>string</code>	Kredensi yang diperlukan untuk memanggil otorisasi, jika ada, dalam bentuk ARN dari peran eksekusi IAM. Misalnya, "arn:aws:iam:::". <i>account-id IAM_role</i>
<code>authorizerPayloadFormatVersion</code>	<code>string</code>	Untuk HTTP APIs, tentukan format data yang dikirimkan API Gateway ke otorisasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Untuk mempelajari selengkapnya, lihat the section called “Versioning the payload format”.

Nama properti	Tipe	Deskripsi
<code>enableSimpleResponses</code>	Boolean	Untuk HTTP APIs, menentukan apakah <code>request authorizer</code> mengembalikan nilai Boolean atau kebijakan IAM. Didukung hanya untuk otorisasi dengan <code>authorizerPayloadFormatVersion</code> dari 2.0. Jika diaktifkan, fungsi Lambda <code>authorizer</code> mengembalikan nilai Boolean. Untuk mempelajari selengkapnya, lihat the section called “Respons fungsi Lambda untuk format 2.0” .
<code>identitySource</code>	string	Daftar ekspresi pemetaan parameter permintaan yang dipisahkan koma sebagai sumber identitas. Berlaku untuk otorisasi <code>request</code> dan <code>jwt</code> tipe saja.
<code>jwtConfiguration</code>	Object	Menentukan penerbit dan audiens untuk otorisasi JWT. Untuk mempelajari selengkapnya, lihat JWTConfiguration di Referensi API API Gateway Versi 2. Hanya didukung untuk HTTP APIs.

Nama properti	Tipe	Deskripsi
identityValidationExpression	string	Ekspresi reguler untuk memvalidasi token sebagai identitas yang masuk. Misalnya, <code>^x- [a-z] +</code> . Didukung hanya untuk TOKEN otorisasi untuk REST APIs.
authorizerResultTtlInSeconds	string	Jumlah detik selama hasil otorisasi di-cache.
providerARNs	Sebuah array dari string	Daftar kumpulan pengguna Amazon Cognito ARNs untuk COGNITO_USER_POOLS

x-amazon-apigateway-authorizer contoh untuk REST APIs

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “token” dan bernama `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
  }
}
}

```

Cuplikan objek operasi OpenAPI berikut menetapkan GET `/http` untuk menggunakan otorisasi Lambda sebelumnya.

```

"/http" : {
  "get" : {
    "responses" : { },
    "security" : [ {
      "test-authorizer" : [ ]
    } ],
    "x-amazon-apigateway-integration" : {
      "type" : "http",
      "responses" : {
        "default" : {
          "statusCode" : "200"
        }
      },
      "httpMethod" : "GET",
      "uri" : "http://api.example.com"
    }
  }
}
}

```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “permintaan”, dengan parameter header tunggal () `auth` sebagai sumber identitas. `securityDefinitions` itu dinamai `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
  }
}
}

```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “permintaan”, dengan satu header (HeaderAuth1) dan satu parameter string kueri sebagai sumber identitas. QueryString1

```

"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
    }
  }
}

```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda API Gateway dari jenis “permintaan”, dengan variabel tahap tunggal () stage sebagai sumber identitas.

```

"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
    }
  }
}

```

Contoh definisi keamanan OpenAPI berikut menentukan kumpulan pengguna Amazon Cognito sebagai otorisasi.

```

"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"
      ]
    }
  }
}

```

Cuplikan objek operasi OpenAPI berikut menyetel GET /http untuk menggunakan kumpulan pengguna Amazon Cognito sebelumnya sebagai otorisasi, tanpa cakupan khusus.

```

"/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 contoh untuk HTTP APIs

Contoh OpenAPI 3.0 berikut membuat otorisasi JWT untuk API HTTP yang menggunakan Amazon Cognito sebagai penyedia identitas, dengan header sebagai sumber identitas. Authorization

```

"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"
    }
  }
}

```

Contoh OpenAPI 3.0 berikut menghasilkan otorisasi JWT yang sama dengan contoh sebelumnya. Namun, contoh ini menggunakan `openIdConnectUrl` properti OpenAPI untuk mendeteksi penerbit secara otomatis. `openIdConnectUrl` harus sepenuhnya terbentuk.

```
"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"
    }
  }
}
```

Contoh berikut membuat authorizer Lambda untuk HTTP API. Authorizer contoh ini menggunakan `Authorization` header sebagai sumber identitasnya. Authorizer menggunakan versi format 2.0 payload, dan mengembalikan nilai Boolean, karena `enableSimpleResponses` diatur ke `true`.

```
"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 properti

Untuk REST APIs, ekstensi ini dapat digunakan untuk menentukan jenis kustom dari otorisasi Lambda. Dalam hal ini, nilainya adalah bentuk bebas. Misalnya, API mungkin memiliki beberapa otorisasi Lambda yang menggunakan skema internal yang berbeda. Anda dapat menggunakan ekstensi ini untuk mengidentifikasi skema internal otorisasi Lambda.

Lebih umum, di HTTP APIs dan REST APIs, itu juga dapat digunakan sebagai cara untuk mendefinisikan otorisasi IAM di beberapa operasi yang berbagi skema keamanan yang sama. Dalam hal ini, istilah tersebut `awsSigv4` adalah istilah yang dicadangkan, bersama dengan istilah apa pun yang diawali oleh `aws`.

[Ekstensi ini berlaku untuk skema keamanan `apiKey` tipe di OpenAPI 2 dan OpenAPI 3.](#)

x-amazon-apigateway-authtype contoh

Contoh OpenAPI 3 berikut mendefinisikan otorisasi IAM di beberapa sumber daya dalam REST API atau 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"
    }
  }
}
}
}

```

Contoh OpenAPI 3 berikut mendefinisikan otorisasi Lambda dengan skema khusus untuk 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"
}

```

Lihat juga

[Authorizer.authType](#)

x-amazon-apigateway-binaryproperty -media-tipe

Menentukan daftar jenis media biner yang akan didukung oleh API Gateway, seperti `application/octet-stream` dan `image/jpeg`. Ekstensi ini adalah array JSON. Ini harus disertakan sebagai ekstensi vendor tingkat atas ke dokumen OpenAPI.

x-amazon-apigateway-binary-media-jenis contoh

Contoh berikut menunjukkan urutan pencarian pengkodean API.

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

x-amazon-apigateway-documentation objek

Mendefinisikan bagian dokumentasi yang akan diimpor ke API Gateway. Objek ini adalah objek JSON yang berisi array dari `DocumentationPart` instance.

Nama properti	Tipe	Deskripsi
documentationParts	Array	Array DocumentationPart instance yang diekspor atau diimpor.
version	String	Pengidentifikasi versi snapshot dari bagian dokumentasi yang diekspor.

x-amazon-apigateway-documentation contoh

Contoh ekstensi API Gateway ke OpenAPI berikut mendefinisikan DocumentationParts instance yang akan diimpor atau diekspor dari API di API Gateway.

```
{ ...
  "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-modus akses

Menentukan mode akses endpoint untuk REST API atau nama domain kustom. Untuk mempelajari lebih lanjut tentang mode akses titik akhir, lihat [Mode akses titik akhir](#).

Contoh x-amazon-apigateway-endpoint-access-mode

Contoh berikut menentukan mode akses endpoint ke ketat.

```
"x-amazon-apigateway-endpoint-access-mode": "STRICT"
```

x-amazon-apigateway-endpoint-konfigurasi objek

Menentukan rincian konfigurasi endpoint untuk API. Ekstensi ini adalah properti yang diperluas dari objek [Operasi OpenAPI](#). Objek ini harus ada di [ekstensi vendor tingkat atas](#) untuk Swagger 2.0. Untuk OpenAPI 3.0, itu harus ada di bawah ekstensi vendor dari objek [Server](#).

Nama properti	Tipe	Deskripsi
<code>disableExecuteApiEndpoint</code>	Boolean	Menentukan apakah klien dapat memanggil API dengan menggunakan titik akhir <code>execute-api</code> default. Secara default, klien dapat memanggil API Anda dengan titik <code>https://{api_id}.execute-api.{region}.amazonaws.com</code> akhir default. Untuk mengharuskan klien menggunakan nama domain khusus untuk menjalankan API Anda, tentukan <code>true</code> .
<code>vpcEndpointIds</code>	Sebuah array dari <code>String</code>	Daftar <code>VpcEndpoint</code> pengidentifikasi yang digunakan untuk membuat catatan alias Route

Nama properti	Tipe	Deskripsi
		53 untuk REST API. Ini hanya didukung untuk REST APIs tipe PRIVATE titik akhir.
ipAddressType	string	Jenis alamat IP yang dapat memanggil API HTTP. Gunakan ipv4 untuk mengizinkan jenis IPv4 alamat untuk memanggil API HTTP. Gunakan dualstack untuk mengizinkan IPv4 dan IPv6 menangani jenis untuk menjalankan API HTTP. Ini hanya didukung untuk HTTP APIs.

x-amazon-apigateway-endpoint-contoh konfigurasi

Contoh berikut mengaitkan titik akhir VPC tertentu ke REST API.

```
"x-amazon-apigateway-endpoint-configuration": {  
  "vpceEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]  
}
```

Contoh berikut menonaktifkan titik akhir default untuk API.

```
"x-amazon-apigateway-endpoint-configuration": {  
  "disableExecuteApiEndpoint": true  
}
```

Contoh berikut menetapkan jenis alamat IP untuk dualstack untuk HTTP API.

```
"x-amazon-apigateway-endpoint-configuration": {  
  "ipAddressType": "dualstack"  
}
```

x-amazon-apigateway-gateway-tanggapan objek

Mendefinisikan respons gateway untuk API sebagai string-to- [GatewayResponse](#)map dari pasangan nilai kunci. Ekstensi ini berlaku untuk struktur OpenAPI tingkat root.

Nama properti	Tipe	Deskripsi
<i>responseType</i>	x-amazon-apigateway-gateway-Responses.gatewayResponse	A GatewayResponse untuk yang ditentukan <i>responseType</i> .

x-amazon-apigateway-gateway-contoh tanggapan

Ekstensi API Gateway berikut ke OpenAPI contoh mendefinisikan [GatewayResponses](#)peta yang berisi dua [GatewayResponse](#)instance—satu untuk tipe dan satu lagi untuk tipe. DEFAULT_4XX 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 objek

Mendefinisikan respons gateway dari jenis respons tertentu, termasuk kode status, parameter respons apa pun yang berlaku, atau templat respons.

Nama properti	Tipe	Deskripsi
<i>responseParameters</i>	x-amazon-apigateway-gateway-Responses.ResponseParameters	Menentukan GatewayResponseParameter , yaitu parameter header. Nilai parameter dapat mengambil nilai parameter permintaan masuk atau nilai kustom statis.
<i>responseTemplates</i>	x-amazon-apigateway-gateway-Responses.ResponseTemplates	Menentukan template pemetaan dari respon gateway. Template tidak diproses oleh mesin VTL.
<i>statusCode</i>	string	Kode status HTTP untuk respons gateway.

x-amazon-apigateway-gateway-Responses.gatewayResponse contoh

Contoh berikut dari ekstensi API Gateway ke OpenAPI mendefinisikan a [GatewayResponse](#) untuk menyesuaikan INVALID_API_KEY respons untuk mengembalikan kode status 456, nilai api-key header permintaan masuk, dan pesan. "Bad api-key"

```
"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 objek

Mendefinisikan string-to-string peta pasangan kunci-nilai untuk menghasilkan parameter respons gateway dari parameter permintaan masuk atau menggunakan string literal. Didukung hanya untuk REST APIs.

Nama properti	Tipe	Deskripsi
gatewayresponse. <i>param-position</i> . <i>param-name</i>	string	<i>param-position</i> bisa header, path, atau querystring . Untuk informasi selengkapnya, lihat Pemetaan parameter untuk REST API di API Gateway .

x-amazon-apigateway-gateway-Responses.ResponseParameters contoh

Contoh ekstensi OpenAPI berikut menunjukkan ekspresi pemetaan parameter

[GatewayResponse](#) respons untuk mengaktifkan dukungan CORS untuk sumber daya pada domain.

`*.example.domain`

```
"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 objek

Mendefinisikan templat [GatewayResponse](#) pemetaan, sebagai string-to-string peta pasangan nilai kunci, untuk respons gateway yang diberikan. Untuk setiap pasangan kunci-nilai, kuncinya adalah tipe konten. Misalnya, “application/json” dan nilainya adalah template pemetaan stringified untuk substitusi variabel sederhana. Template GatewayResponse pemetaan tidak diproses oleh mesin [Velocity Template Language \(VTL\)](#).

Nama properti	Tipe	Deskripsi
<i>content-type</i>	string	Template pemetaan GatewayResponse tubuh yang hanya mendukung substitusi variabel sederhana untuk menyesuaikan badan respons gateway.

x-amazon-apigateway-gateway-Responses.responseTemplates contoh

Contoh ekstensi OpenAPI berikut menunjukkan template [GatewayResponse](#) pemetaan untuk menyesuaikan respons kesalahan yang dihasilkan oleh gateway API ke dalam format khusus aplikasi.

```
"responseTemplates": {
  "application/json": "{ \"message\": $context.error.messageString, \"type\": $context.error.responseType, \"statusCode\": '488' }"
}
```

Contoh ekstensi OpenAPI berikut menunjukkan template [GatewayResponse](#) pemetaan untuk mengganti respons kesalahan yang dihasilkan oleh gateway API dengan pesan kesalahan statis.

```
"responseTemplates": {
  "application/json": "{ \"message\": 'API-specific errors' }"
}
```

x-amazon-apigateway-importexport-versi

Menentukan versi algoritma impor dan ekspor API Gateway untuk HTTP APIs. Saat ini, satu-satunya nilai yang didukung adalah 1.0. Untuk mempelajari lebih lanjut, lihat [ExportVersion](#) di Referensi API Gateway Versi 2 API.

x-amazon-apigateway-importexport-versi contoh

Contoh berikut menetapkan versi impor dan ekspor ke 1.0.

```
{
  "openapi": "3.0.1",
  "x-amazon-apigateway-importexport-version": "1.0",
  "info": { ...
```

x-amazon-apigateway-integration objek

Menentukan rincian integrasi backend yang digunakan untuk metode ini. Ekstensi ini adalah properti yang diperluas dari objek [Operasi OpenAPI](#). Hasilnya adalah objek [integrasi API Gateway](#).

Nama properti	Tipe	Deskripsi
cacheKeyParameters	Sebuah array dari string	Daftar parameter permintaan yang nilainya akan di-cache.
cacheNamespace	string	Grup tag spesifik API dari parameter yang di-cache terkait.
connectionId	string	ID a VpcLink untuk integrasi pribadi.
connectionType	string	Jenis koneksi integrasi. Nilai yang valid adalah "VPC_LINK" untuk integrasi pribadi atau "INTERNET", sebaliknya a.

Nama properti	Tipe	Deskripsi
credentials	string	Untuk kredensial berbasis peran AWS IAM, tentukan ARN dari peran IAM yang sesuai. Jika tidak ditentukan, kredensial default ke izin berbasis sumber daya yang harus ditambahkan secara manual agar API dapat mengakses sumber daya. Untuk informasi selengkapnya, lihat Memberikan Izin Menggunakan Kebijakan Sumber Daya. Catatan: Saat menggunakan kredensial IAM, pastikan titik akhir Regional AWS STS diaktifkan untuk Wilayah tempat API ini digunakan untuk kinerja terbaik.

Nama properti	Tipe	Deskripsi
contentHandling	string	Minta jenis konversi pengkodean muatan. Nilai yang valid adalah 1)CONVERT_TO_TEXT , untuk mengubah muatan biner menjadi string yang dikodekan base64 atau mengubah muatan teks menjadi string yang utf-8 dikodekan atau melewati muatan teks secara asli tanpa modifikasi, dan 2)CONVERT_TO_BINARY , untuk mengubah muatan teks menjadi gumpalan yang diterjemahkan base64 atau melewati muatan biner secara asli tanpa modifikasi.
httpMethod	string	Metode HTTP yang digunakan dalam permintaan integrasi . Untuk pemanggilan fungsi Lambda, nilainya harus. POST
integrationSubtype	string	Menentukan subtype integrasi untuk integrasi AWS layanan. Hanya didukung untuk HTTP APIs. Untuk subtype integrasi yang didukung, lihat the section called “AWS referensi integrasi layanan” .

Nama properti	Tipe	Deskripsi
integrationTarget	string	Pendengar ALB atau NLB untuk mengirim permintaan ke. Hanya didukung untuk integrasi pribadi yang menggunakan tautan VPC V2. Untuk informasi selengkapnya, lihat the section called “Tautan VPC V2” .
passthroughBehavior	string	Menentukan bagaimana payload permintaan jenis konten yang tidak dipetakan diteruskan melalui permintaan integrasi tanpa modifikasi. Nilai yang didukung adalah <code>when_no_templates</code> , <code>when_no_match</code> , dan <code>never</code> . Untuk informasi selengkapnya, lihat Integration.PassThroughBehavior .
payloadFormatVersion	string	Menentukan format muatan yang dikirim ke integrasi. Diperlukan untuk HTTP APIs. Untuk HTTP APIs, nilai yang didukung untuk integrasi proxy Lambda adalah <code>1.0</code> dan <code>2.0</code> . Untuk semua integrasi lainnya, <code>1.0</code> adalah satu-satunya nilai yang didukung. Untuk mempelajari selengkapnya, lihat the section called “AWS Lambda integrasi” dan the section called “AWS referensi integrasi layanan” .

Nama properti	Tipe	Deskripsi
<code>requestParameters</code>	x-amazon-apigateway-integrationObjek. RequestParameters	Untuk REST APIs, menentukan pemetaan dari parameter permintaan metode ke parameter permintaan integrasi. Parameter permintaan yang didukung adalah <code>queryString</code>, <code>path</code>, <code>header</code>, dan <code>body</code>. Untuk HTTP APIs, parameter permintaan adalah peta nilai kunci yang menentukan parameter yang diteruskan ke <code>AWS_PROXY</code> integrasi dengan yang ditentukan. <code>integrationSubtype</code> Anda dapat memberikan nilai statis, atau data permintaan peta, variabel tahap, atau variabel konteks yang dievaluasi pada saat runtime. Untuk mempelajari selengkapnya, lihat the section called “AWS integrasi layanan”.
<code>requestTemplates</code>	x-amazon-apigateway-integrationObjek. RequestTemplates	Memetakan template untuk payload permintaan tipe MIME tertentu.

Nama properti	Tipe	Deskripsi
responses	x-amazon-apigateway-integration.response objek	Mendefinisikan respons metode dan menentukan pemetaan parameter atau pemetaan muatan yang diinginkan dari respons integrasi hingga respons metode.
responseTransferMode	string	Mode transfer respons integrasi. Gunakan BUFFERED agar API Gateway menunggu untuk menerima respons lengkap sebelum memulai transmisi. Gunakan STREAM agar API Gateway mengirim sebagian tanggapan kembali ke klien saat tersedia. Untuk mempelajari selengkapnya, lihat the section called “Mengalirkan respons integrasi untuk integrasi proksi Anda” .
timeoutInMillis	integer	Batas waktu integrasi antara 50 ms dan 29.000 ms.

Nama properti	Tipe	Deskripsi
type	string	Jenis integrasi dengan backend yang ditentukan. Nilai yang valid adalah: • <code>http</code> atau <code>http_proxy</code> , untuk integrasi dengan backend HTTP. • <code>aws_proxy</code> , untuk integrasi dengan fungsi AWS Lambda. • <code>aws</code>, untuk integrasi dengan fungsi AWS Lambda atau AWS layanan lain, seperti Amazon DynamoDB, Amazon Simple Notification Service, atau Amazon Simple Queue Service. • <code>mock</code>, untuk integrasi dengan API Gateway tanpa memanggil backend apa pun. Untuk informasi selengkap nya tentang tipe integrasi, lihat integration:type.
tlsConfig	the section called “x-amazon-apigateway-integration.tlsConfig”	Menentukan konfigurasi TLS untuk integrasi.

Nama properti	Tipe	Deskripsi
uri	string	URI endpoint dari backend. Untuk integrasi aws tipe, ini adalah nilai ARN. Untuk integrasi HTTP, ini adalah URL dari titik akhir HTTP termasuk http skema https or.

x-amazon-apigateway-integration contoh

Untuk HTTP APIs, Anda dapat menentukan integrasi di bagian komponen definisi OpenAPI Anda. Untuk mempelajari selengkapnya, lihat [x-amazon-apigateway-integrations objek](#).

```
"x-amazon-apigateway-integration": {
  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

Contoh berikut membuat integrasi dengan fungsi Lambda. Untuk tujuan demonstrasi, contoh templat pemetaan yang ditunjukkan dalam requestTemplates dan responseTemplates contoh di bawah ini diasumsikan berlaku untuk muatan berformat JSON berikut: { "name": "value_1", "key": "value_2", "redirect": { "url" : "..."} } untuk menghasilkan output JSON dari atau output XML. dari. { "stage": "value_1", "user-id": "value_2" }
<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> "
```


x-amazon-apigateway-integrations objek

Mendefinisikan kumpulan integrasi. Anda dapat menentukan integrasi di bagian komponen definisi OpenAPI Anda, dan menggunakan kembali integrasi untuk beberapa rute. Hanya didukung untuk HTTP APIs.

Nama properti	Tipe	Deskripsi
<i>integration</i>	x-amazon-apigateway-integration objek	Kumpulan objek integrasi.

x-amazon-apigateway-integrations contoh

Contoh berikut membuat API HTTP yang mendefinisikan dua integrasi, dan referensi integrasi dengan menggunakan `$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-integrationObjek. RequestTemplates

Menentukan template pemetaan untuk payload permintaan dari jenis MIME tertentu.

Nama properti	Tipe	Deskripsi
<i>MIME type</i>	string	Contoh tipe MIME adalah <code>application/json</code> . Untuk informasi tentang membuat template pemetaan, lihat the section called “Pemetaan transformasi template” .

x-amazon-apigateway-integrationContoh. RequestTemplates

Contoh berikut menetapkan template pemetaan untuk payload permintaan tipe `application/json` dan `application/xml` MIME.

```

"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-integrationObjek. RequestParameters

Untuk REST APIs, menentukan pemetaan dari parameter permintaan metode bernama ke parameter permintaan integrasi. Parameter permintaan metode harus didefinisikan sebelum direferensikan.

Untuk HTTP APIs, menentukan parameter yang diteruskan ke AWS_PROXY integrasi dengan yang ditentukan. `integrationSubtype`

Nama properti	Tipe	Deskripsi
<code>integration.request. t. <i><param-type></i> <i>pe> .<param-name></i></code>	string	Untuk REST APIs, nilainya biasanya merupakan parameter permintaan metode standar dari <code>method.request.<i><param-type></i> <i>pe> .<param-name></i></code> format, di mana <code><param-type></code> <code>pe></code> bisa <code>querystring</code> , <code>pathheader</code> , atau <code>body</code> . Namun, <code>\$context.VARIABLE_NAME</code> , <code>\$stageVariables.VARIABLE_NAME</code> , dan <code>STATIC_VALUE</code> juga valid. Untuk body parameter, <code><param-name></code> adalah ekspresi jalur JSON tanpa <code>\$</code> awalan.
<i>parameter</i>	string	Untuk HTTP APIs, parameter permintaan adalah peta nilai kunci yang menentukan

Nama properti	Tipe	Deskripsi
		parameter yang diteruskan ke AWS_PROXY integrasi dengan yang ditentukan. integrationSubtype Anda dapat memberikan nilai statis, atau data permintaan peta, variabel tahap, atau variabel konteks yang dievaluasi pada saat runtime. Untuk mempelajari selengkapnya, lihat the section called “AWS integrasi layanan” .

Contoh x-amazon-apigateway-integration.requestParameters

Contoh pemetaan parameter permintaan berikut menerjemahkan parameter query (version), header (), dan path (x-user-idservice) permintaan metode ke query permintaan integrasi (stage), header (), dan parameter jalur (x-userid), masing-masing.

Note

Jika Anda membuat sumber daya melalui OpenAPI atau CloudFormation, nilai statis harus disertakan dalam tanda kutip tunggal.
Untuk menambahkan nilai ini dari konsol, masukkan application/json di dalam kotak, tanpa tanda kutip.

```
"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.response objek

Mendefinisikan respons metode dan menentukan pemetaan parameter atau pemetaan muatan dari respons integrasi ke respons metode.

Nama properti	Tipe	Deskripsi
<i>Response status pattern</i>	x-amazon-apigateway-integration.response objek	Baik ekspresi reguler yang digunakan untuk mencocokkan respons integrasi dengan respons metode, atau default untuk menangkap respons apa pun yang belum Anda konfigurasi. Untuk integrasi HTTP, regex berlaku untuk kode status respons integrasi. Untuk pemanggilan Lambda, regex berlaku untuk errorMessage bidang objek informasi kesalahan yang dikembalikan oleh AWS Lambda sebagai badan respons kegagalan saat eksekusi fungsi Lambda melempar pengecualian.

 **Note**
Nama *Response status pattern* properti mengacu pada kode status respons atau ekspresi reguler yang menggambarkan sekelompok kode status respons. Itu

Nama properti	Tipe	Deskripsi
		tidak sesuai dengan pengenalan Integrasi onResponse sumber daya apa pun di API Gateway REST API.

Contoh x-amazon-apigateway-integration.responses

Contoh berikut menunjukkan daftar tanggapan dari 2xx dan 302 tanggapan. Untuk responsnya, 2xx respons metode dipetakan dari muatan respons integrasi tipe MIME `application/json` atau `application/xml` MIME. Respons ini menggunakan template pemetaan yang disediakan. Untuk 302 respon, respon metode mengembalikan Location header yang nilainya berasal dari `redirect.url` properti pada payload respon integrasi.

```
"responses" : {
  "2\\d{2}" : {
    "statusCode" : "200",
    "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"
    }
  }
}
```

x-amazon-apigateway-integration.response objek

Mendefinisikan respons dan menentukan pemetaan parameter atau pemetaan muatan dari respons integrasi ke respons metode.

Nama properti	Tipe	Deskripsi
statusCode	string	Kode status HTTP untuk respons metode; misalnya, "200". Ini harus sesuai dengan respons yang cocok di bidang Operasi responses OpenAPI .
responseTemplates	x-amazon-apigateway-integrationObjek. ResponseTemplates	Menentukan templat pemetaan khusus tipe MIME untuk muatan respons.
responseParameters	x-amazon-apigateway-integration.ResponseParameters objek	Menentukan pemetaan parameter untuk respon. Hanya body parameter header dan respons integrasi yang dapat dipetakan ke header parameter metode.
contentHandling	string	Jenis konversi pengkodean muatan respons. Nilai yang valid adalah 1) <code>CONVERT_TO_TEXT</code> , untuk mengubah muatan biner menjadi string yang dikodekan base64 atau mengubah muatan teks menjadi string yang <code>utf-8</code> dikodekan atau melewati muatan teks secara asli tanpa modifikasi, dan 2) <code>CONVERT_TO_BINARY</code> ,

Nama properti	Tipe	Deskripsi
		untuk mengubah muatan teks menjadi gumpalan yang diterjemahkan base64 atau melewati muatan biner secara asli tanpa modifikasi.

Contoh `x-amazon-apigateway-integration.response`

Contoh berikut mendefinisikan 302 respons untuk metode yang memperoleh muatan tipe `application/json` atau `application/xml` MIME dari backend. Respons menggunakan templat pemetaan yang disediakan dan mengembalikan URL pengalihan dari respons integrasi di header metode. `Location`

```
{
  "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-integrationObjek. ResponseTemplates`

Menentukan template pemetaan untuk payload respon dari jenis MIME tertentu.

Nama properti	Tipe	Deskripsi
<i>MIME type</i>	string	Menentukan template pemetaan untuk mengubah

Nama properti	Tipe	Deskripsi
		badan respons integrasi ke badan respons metode untuk tipe MIME tertentu. Untuk informasi tentang membuat template pemetaan, lihat the section called “Pemetaan transformasi template”. Contoh dari <i>MIME type</i> adalah <code>application/json</code> .

x-amazon-apigateway-integrationContoh .responseTemplate

Contoh berikut menetapkan template pemetaan untuk payload permintaan tipe `application/json` dan `application/xml` MIME.

```
"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 objek

Menentukan pemetaan dari parameter respon metode integrasi untuk parameter respon metode. Anda dapat memetakan header body, atau nilai statis ke header jenis respons metode. Didukung hanya untuk REST APIs.

Nama properti	Tipe	Deskripsi
<code>method.response.header. <i><param-name></i></code>	string	Nilai parameter bernama dapat diturunkan dari header

Nama properti	Tipe	Deskripsi
		dan body jenis parameter respons integrasi.

Contoh `x-amazon-apigateway-integration.responseParameters`

Berikut contoh peta body dan header parameter dari respon integrasi untuk dua header parameter dari respon metode.

```
"responseParameters" : {
  "method.response.header.Location" : "integration.response.body.redirect.url",
  "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

`x-amazon-apigateway-integrationObjek .tlsConfig`

Menentukan konfigurasi TLS untuk integrasi.

Nama properti	Tipe	Deskripsi
<code>insecureSkipVerification</code>	Boolean	Didukung hanya untuk REST APIs. Menentukan apakah API Gateway melewati verifikasi bahwa sertifikat untuk titik akhir integrasi dikeluarkan oleh otoritas sertifikat yang didukung . Ini tidak disarankan, tetapi memungkinkan Anda untuk menggunakan sertifikat yang ditandatangani oleh otoritas sertifikat swasta, atau sertifikat yang ditandatangani sendiri. Jika diaktifkan, API Gateway masih melakukan

Nama properti	Tipe	Deskripsi
		validasi sertifikat dasar, termasuk memeriksa tanggal kedaluwarsa sertifikat, nama host, dan keberadaan otoritas sertifikat root. Sertifikat root milik otoritas swasta harus memenuhi kendala berikut: • ekstensi x509 keyUsage harus memiliki. keyCertSi gn • ekstensi x509 basicConstraints harus memiliki. CA:TRUE Didukung hanya untuk HTTP dan HTTP_PROXY integrasi.  Warning Pengaktifan insecureSkipVerification tidak disarankan, terutama untuk integrasi dengan titik akhir HTTPS publik. Jika Anda mengaktifkan insecureSkipVerification, Anda meningkatkan risiko man-in-the-middle serangan.

Nama properti	Tipe	Deskripsi
<code>serverNameToVerify</code>	<code>string</code>	Hanya didukung untuk integrasi pribadi HTTP API. Jika Anda menentukan nama server, API Gateway menggunakannya untuk memverifikasi nama host pada sertifikat integrasi. Nama server juga termasuk dalam jabat tangan TLS untuk mendukung Indikasi Nama Server (SNI) atau hosting virtual.

x-amazon-apigateway-integrationContoh .tlsConfig

Contoh OpenAPI 3.0 berikut memungkinkan integrasi `insecureSkipVerification` proxy HTTP REST API.

```
"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",
}
```

Contoh OpenAPI 3.0 berikut menentukan `serverNameToVerify` untuk integrasi pribadi HTTP API.

```
"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-ukuran kompresi

Menentukan ukuran kompresi minimum untuk REST API. Untuk mengaktifkan kompresi, tentukan bilangan bulat antara 0 dan 10485760. Untuk mempelajari selengkapnya, lihat [Kompresi muatan untuk REST APIs di API Gateway](#).

x-amazon-apigateway-minimum-contoh ukuran kompresi

Contoh berikut menentukan ukuran kompresi minimum 5242880 byte untuk REST API.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Menentukan kebijakan sumber daya untuk REST API. Untuk mempelajari lebih lanjut tentang kebijakan sumber daya, lihat [Kontrol akses ke REST API dengan kebijakan sumber daya API Gateway](#). Untuk contoh kebijakan sumber daya, lihat [Contoh kebijakan sumber daya API Gateway](#).

Contoh x-amazon-apigateway-policy

Contoh berikut menentukan kebijakan sumber daya untuk REST API. Kebijakan sumber daya menolak (memblokir) lalu lintas masuk ke API dari blok alamat IP sumber tertentu. Saat diimpor, "execute-api:/*" dikonversi ke `arn:aws:execute-api:region:account-id:api-id/*`, menggunakan Wilayah saat ini, ID AWS akun Anda, dan ID REST API saat ini.

```
"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-requestproperty -validator

Menentukan validator permintaan, dengan mereferensikan [x-amazon-apigateway-request-validator objek](#) peta, untuk mengaktifkan validasi permintaan pada API yang mengandung atau metode. *request_validator_name* Nilai ekstensi ini adalah string JSON.

Ekstensi ini dapat ditentukan pada tingkat API atau di tingkat metode. Validator API-level berlaku untuk semua metode kecuali jika diganti oleh validator tingkat metode.

Contoh x-amazon-apigateway-request-validator

Contoh berikut menerapkan validator basic permintaan di API level saat menerapkan validator parameter-only permintaan pada permintaan. POST /validation

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-validator objek

Mendefinisikan validator permintaan yang didukung untuk API yang berisi sebagai peta antara nama validator dan aturan validasi permintaan terkait. Ekstensi ini berlaku untuk REST API.

Nama properti	Tipe	Deskripsi
<i>request_validator_name</i>	x-amazon-apigateway-request-Validators.requestValidator objek	Menentukan aturan validasi yang terdiri dari validator bernama. Contoh: <pre>"basic" : { "validate RequestBody" : true, "validate RequestParameters" : true },</pre> Untuk menerapkan validator ini ke metode tertentu,

Nama properti	Tipe	Deskripsi
		referensi nama validator (basic) sebagai nilai properti. x-amazon-apigateway-request-properti -validator

Contoh x-amazon-apigateway-request-validators

Contoh berikut menunjukkan sekumpulan validator permintaan untuk API sebagai peta antara nama validator dan aturan validasi permintaan terkait.

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 objek

Menentukan aturan validasi validator permintaan sebagai bagian dari definisi peta. [x-amazon-apigateway-request-validator objek](#)

Nama properti	Tipe	Deskripsi
<code>validateRequestBody</code>	Boolean	Menentukan apakah untuk memvalidasi badan permintaan (<code>true</code>) atau tidak (<code>false</code>).
<code>validateRequestParameters</code>	Boolean	Menentukan apakah untuk memvalidasi parameter permintaan yang diperlukan (<code>true</code>) atau tidak (<code>false</code>).

Contoh `x-amazon-apigateway-request-validators.requestValidator`

Contoh berikut menunjukkan validator permintaan parameter-only:

```
"params-only": {
  "validateRequestBody" : false,
  "validateRequestParameters" : true
}
```

x-amazon-apigateway-security-kebijakan

Menentukan kebijakan keamanan untuk REST API. Jika Anda membuat kebijakan keamanan yang dimulai `"SecurityPolicy_"`, Anda juga harus mengatur [mode akses titik akhir](#). Untuk mempelajari lebih lanjut tentang kebijakan keamanan, lihat [Kebijakan keamanan untuk REST API di API Gateway](#).

Contoh `x-amazon-apigateway-security-policy`

Contoh berikut menentukan `SecurityPolicy_TLS13_1_3_2025_0` untuk REST API.

```
"x-amazon-apigateway-security-policy": "SecurityPolicy_TLS13_1_3_2025_09"
```

x-amazon-apigateway-tag-nilai properti

Menentukan nilai [AWS tag](#) untuk HTTP API. Anda dapat menggunakan `x-amazon-apigateway-tag-value` properti sebagai bagian dari [objek tag OpenAPI tingkat root untuk menentukan AWS](#)

[tag](#) untuk API HTTP. Jika Anda menentukan nama tag tanpa `x-amazon-apigateway-tag-value` properti, API Gateway akan membuat tag dengan string kosong untuk nilai.

Untuk mempelajari lebih lanjut tentang penandaan, lihat [Menandai sumber daya API Gateway Anda](#).

Nama properti	Tipe	Deskripsi
<code>name</code>	String	Menentukan kunci tag.
<code>x-amazon-apigateway-tag-value</code>	String	Menentukan nilai tag.

Contoh `x-amazon-apigateway-tag-value`

Contoh berikut menentukan dua tag untuk HTTP API:

- “Pemilik”: “Admin”
- “Produk”: “”

```
"tags": [  
  {  
    "name": "Owner",  
    "x-amazon-apigateway-tag-value": "Admin"  
  },  
  {  
    "name": "Prod"  
  }  
]
```

Keamanan di Amazon API Gateway

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama model](#) menggambarkan ini sebagai keamanan cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program AWS Kepatuhan Program AWS Kepatuhan](#) . Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon API Gateway, lihat [AWS layanan dalam cakupan berdasarkan AWS layanan program kepatuhan](#) .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan API Gateway. Topik berikut menunjukkan cara mengonfigurasi API Gateway untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya API Gateway Anda.

Untuk informasi selengkapnya, lihat [Gambaran Umum Keamanan Amazon API Gateway](#).

Topik

- [Perlindungan data di Amazon API Gateway](#)
- [Manajemen identitas dan akses untuk Amazon API Gateway](#)
- [Pencatatan dan pemantauan di Amazon API Gateway](#)
- [Validasi kepatuhan untuk Amazon API Gateway](#)
- [Ketahanan di Amazon API Gateway](#)
- [Keamanan infrastruktur di Amazon API Gateway](#)
- [Analisis kerentanan di Amazon API Gateway](#)
- [Praktik terbaik keamanan di Amazon API Gateway](#)

Perlindungan data di Amazon API Gateway

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di Amazon API Gateway. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat [FAQ Privasi Data AWS](#) . Untuk informasi tentang perlindungan data di Eropa, lihat [Pusat Peraturan Umum Perlindungan Data \(GDPR\)](#).

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensi dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan API Gateway atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan

atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi data di Amazon API Gateway

Perlindungan data mengacu pada melindungi data saat transit (saat bepergian ke dan dari API Gateway) dan saat istirahat (saat disimpan di AWS).

Enkripsi data saat istirahat di Amazon API Gateway

Jika Anda memilih untuk mengaktifkan caching untuk REST API, Anda dapat mengaktifkan enkripsi cache. Untuk mempelajari selengkapnya, lihat [Pengaturan cache untuk REST APIs di API Gateway](#).

Untuk informasi selengkapnya tentang perlindungan data, lihat postingan blog [Model Tanggung Jawab Bersama AWS dan GDPR](#) di Blog Keamanan AWS .

Enkripsi dan dekripsi kunci pribadi sertifikat Anda

Saat Anda membuat nama domain khusus untuk pribadi APIs, sertifikat ACM dan kunci pribadi Anda dienkripsi menggunakan kunci KMS AWS terkelola yang memiliki alias `aws/acm`. Anda dapat melihat ID kunci dengan alias ini di AWS KMS konsol di bawah kunci AWS terkelola.

API Gateway tidak langsung mengakses sumber daya ACM Anda. Ini menggunakan AWS TLS Connection Manager untuk mengamankan dan mengakses kunci pribadi untuk sertifikat Anda. Saat Anda menggunakan sertifikat ACM untuk membuat nama domain kustom API Gateway untuk pribadi APIs, API Gateway mengaitkan sertifikat Anda dengan AWS TLS Connection Manager. Ini dilakukan dengan membuat hibah AWS KMS terhadap Kunci AWS Terkelola Anda dengan awalan `aws/acm`. Hibah adalah instrumen kebijakan yang memungkinkan TLS Connection Manager untuk menggunakan kunci KMS dalam operasi kriptografi. Hibah ini memungkinkan prinsipal penerima hibah (TLS Connection Manager) untuk memanggil operasi hibah yang ditentukan pada kunci KMS untuk mendekripsi kunci pribadi sertifikat Anda. TLS Connection Manager kemudian menggunakan sertifikat dan kunci pribadi yang didekripsi (plaintext) untuk membuat koneksi aman (sesi SSL/TLS) dengan klien layanan API Gateway. Ketika sertifikat dipisahkan dari nama domain kustom API Gateway untuk pribadi APIs, hibah akan dihentikan.

Jika Anda ingin menghapus akses ke kunci KMS, kami sarankan Anda mengganti atau menghapus sertifikat dari layanan menggunakan Konsol Manajemen AWS atau `update-service` perintah di AWS CLI.

Konteks enkripsi untuk API Gateway

[Konteks enkripsi](#) adalah kumpulan opsional pasangan kunci-nilai yang berisi informasi kontekstual tentang apa kunci pribadi Anda mungkin digunakan untuk. AWS KMS mengikat konteks enkripsi ke data terenkripsi dan menggunakannya sebagai data otentikasi tambahan untuk mendukung enkripsi yang diautentikasi.

Ketika kunci TLS Anda digunakan dengan API Gateway dan TLS Connection manager, nama layanan API Gateway Anda disertakan dalam konteks enkripsi yang digunakan untuk mengenkripsi kunci Anda saat istirahat. Anda dapat memverifikasi nama domain kustom API Gateway yang digunakan sertifikat dan kunci pribadi Anda dengan melihat konteks enkripsi di CloudTrail log Anda seperti yang ditunjukkan di bagian berikutnya, atau dengan melihat tab Sumber Daya Terkait di konsol ACM.

Untuk mendekripsi data, konteks enkripsi yang sama disertakan dalam permintaan. API Gateway menggunakan konteks enkripsi yang sama di semua operasi kriptografi AWS KMS, di mana kuncinya `aws:apigateway:arn` dan nilainya adalah Nama Sumber Daya Amazon (ARN) dari sumber daya API Gateway. `PrivateDomainName`

Contoh berikut menunjukkan konteks enkripsi dalam output dari operasi seperti `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"
```

Enkripsi data dalam perjalanan di Amazon API Gateway

Yang APIs dibuat dengan Amazon API Gateway hanya mengekspos titik akhir HTTPS. API Gateway tidak mendukung titik akhir (HTTP) yang tidak terenkripsi.

API Gateway mengelola sertifikat untuk `execute-api` titik akhir default. Jika Anda mengonfigurasi nama domain kustom, [Anda menentukan sertifikat untuk nama domain](#). Sebagai praktik terbaik, jangan [menyematkan sertifikat](#).

Untuk keamanan yang lebih baik, Anda dapat memilih versi protokol Transport Layer Security (TLS) minimum yang akan diterapkan untuk domain kustom API Gateway Anda. WebSocket APIs dan APIs dukungan HTTP hanya TLS 1.2. Untuk mempelajari selengkapnya, lihat [Pilih kebijakan keamanan untuk domain kustom Anda di API Gateway](#).

Anda juga dapat mengatur CloudFront distribusi Amazon dengan sertifikat SSL khusus di akun Anda dan menggunakannya dengan Regional APIs. Anda kemudian dapat mengonfigurasi kebijakan keamanan untuk CloudFront distribusi dengan TLS 1.1 atau lebih tinggi berdasarkan persyaratan keamanan dan kepatuhan Anda.

Untuk informasi selengkapnya tentang perlindungan data, lihat [Lindungi REST Anda APIs di API Gateway](#) dan posting blog [Model Tanggung Jawab AWS Bersama dan GDPR](#) di Blog AWS Keamanan.

Privasi lalu lintas antarjaringan

Menggunakan Amazon API Gateway, Anda dapat membuat REST pribadi APIs yang hanya dapat diakses dari Amazon Virtual Private Cloud (VPC). VPC menggunakan antarmuka [VPC endpoint, yang merupakan antarmuka jaringan endpoint](#) yang Anda buat di VPC Anda. Dengan menggunakan [kebijakan sumber daya](#), Anda dapat mengizinkan atau menolak akses ke API dari titik akhir yang dipilih VPCs dan VPC, termasuk di seluruh akun. AWS Setiap endpoint dapat digunakan untuk mengakses beberapa private APIs. Anda juga dapat menggunakan Direct Connect untuk membuat sambungan dari jaringan lokal ke Amazon VPC dan mengakses API pribadi Anda melalui koneksi tersebut. Dalam semua kasus, lalu lintas ke API pribadi Anda menggunakan koneksi aman dan tidak meninggalkan jaringan Amazon; itu terisolasi dari internet publik. Untuk mempelajari informasi lebih lanjut, lihat [the section called “REST pribadi APIs”](#).

Manajemen identitas dan akses untuk Amazon API Gateway

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya API Gateway. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Cara kerja Amazon API Gateway dengan IAM](#)
- [Contoh kebijakan berbasis identitas Amazon API Gateway](#)
- [Contoh kebijakan berbasis sumber daya Amazon API Gateway](#)
- [Memecahkan masalah identitas dan akses Amazon API Gateway](#)
- [Menggunakan peran terkait layanan untuk API Gateway](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat [Memecahkan masalah identitas dan akses Amazon API Gateway](#))
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat [Cara kerja Amazon API Gateway dengan IAM](#))
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat [Contoh kebijakan berbasis identitas Amazon API Gateway](#))

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensial dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), otentikasi masuk tunggal, atau kredensial. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat [Cara masuk ke Akun AWS Anda](#) dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat [Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara](#) di Panduan Pengguna IAM.

[Grup IAM](#) menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan [beralih dari pengguna ke peran IAM \(konsol\)](#) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat [Pilih antara kebijakan terkelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus [menentukan principal](#) dalam kebijakan berbasis sumber daya.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat [Ringkasan daftar kontrol akses \(ACL\)](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat [Kebijakan kontrol layanan](#) dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Cara kerja Amazon API Gateway dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke API Gateway, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan API Gateway. Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja API Gateway dan AWS layanan lainnya dengan IAM, lihat [AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM](#).

Topik

- [Kebijakan berbasis identitas API Gateway](#)
- [Kebijakan berbasis sumber daya API Gateway](#)
- [Otorisasi berdasarkan tag API Gateway](#)
- [Peran IAM API Gateway](#)

Kebijakan berbasis identitas API Gateway

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya mana yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. API Gateway mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk informasi selengkapnya tentang tindakan, sumber daya, dan kunci kondisi khusus Gateway API, lihat Kunci [tindakan, sumber daya, dan kondisi untuk Pengelolaan dan Tindakan Amazon API Gateway, sumber daya, dan kunci kondisi untuk Amazon API Gateway](#) Management V2. Untuk informasi tentang semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi Elemen Kebijakan IAM JSON](#) di Panduan Pengguna IAM.

Contoh berikut menunjukkan kebijakan berbasis identitas yang memungkinkan pengguna untuk membuat atau memperbarui hanya REST pribadi. APIs Untuk contoh lainnya, lihat [the section called "Contoh kebijakan berbasis identitas"](#).

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/*"
]
}
```

Tindakan

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan.

Tindakan kebijakan di API Gateway menggunakan awalan berikut sebelum tindakan: `apigateway:`. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. API Gateway mendefinisikan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

ActionEkspresi API-managing memiliki format `apigateway:action`, di mana `action` salah satu tindakan API Gateway berikut: GET, POST, PUT, DELETE, PATCH (untuk memperbarui sumber daya), atau *, yang merupakan semua tindakan sebelumnya.

Beberapa contoh Action ekspresi meliputi:

- **apigateway:*** untuk semua tindakan API Gateway.
- **apigateway:GET** hanya untuk tindakan GET di API Gateway.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut menggunakan koma seperti berikut:

```
"Action": [
    "apigateway:action1",
    "apigateway:action2"
```

Untuk informasi tentang kata kerja HTTP yang akan digunakan untuk operasi API Gateway tertentu, lihat Referensi [API Amazon API Gateway Versi 1 \(REST APIs\)](#) dan Referensi [API Amazon API Gateway Versi 2](#) (WebSocket dan HTTP APIs).

Untuk informasi selengkapnya, lihat [the section called “Contoh kebijakan berbasis identitas”](#).

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON *Resource* menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": ""
```

Sumber daya API Gateway memiliki format ARN berikut:

```
arn:aws:apigateway:region::resource-path-specifier
```

Misalnya, untuk menentukan REST API dengan id *api-id* dan sub-sumber dayanya, seperti otorisasi dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Untuk menentukan semua REST APIs dan sub-sumber daya milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Untuk daftar jenis sumber daya API Gateway dan jenisnya ARNs, lihat [Referensi API Gateway Amazon Resource Name \(ARN\)](#).

Kunci syarat

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan [operator kondisi](#), misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat [kunci konteks kondisi AWS global](#) di Panduan Pengguna IAM.

API Gateway mendefinisikan kumpulan kunci kondisinya sendiri dan juga mendukung penggunaan beberapa kunci kondisi global. Untuk daftar kunci kondisi API Gateway, lihat [Kunci Kondisi untuk Amazon API Gateway](#) di Panduan Pengguna IAM. Untuk informasi tentang tindakan dan sumber daya yang dapat Anda gunakan dengan kunci kondisi, lihat [Tindakan yang Ditentukan oleh Amazon API Gateway](#).

Untuk informasi tentang penandaan, termasuk kontrol akses berbasis atribut, lihat. [Penandaan](#)

Contoh

Untuk contoh kebijakan berbasis identitas API Gateway, lihat. [Contoh kebijakan berbasis identitas Amazon API Gateway](#)

Kebijakan berbasis sumber daya API Gateway

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya API Gateway dan dalam kondisi apa. API Gateway mendukung kebijakan izin berbasis sumber daya untuk REST APIs Anda menggunakan kebijakan sumber daya untuk mengontrol siapa yang dapat menjalankan REST API. Untuk informasi selengkapnya, lihat [the section called “Menggunakan kebijakan sumber daya API Gateway”](#).

Contoh

Untuk contoh kebijakan berbasis sumber daya API Gateway, lihat. [Contoh kebijakan sumber daya API Gateway](#)

Otorisasi berdasarkan tag API Gateway

Anda dapat melampirkan tag ke resource API Gateway atau meneruskan tag dalam permintaan ke API Gateway. Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di [elemen kondisi](#) dari kebijakan menggunakan kunci kondisi `apigateway:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang menandai resource API Gateway, lihat [the section called “Kontrol Akses Berbasis Atribut”](#).

Untuk contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat. [Menggunakan tag untuk mengontrol akses ke sumber daya API API Gateway REST API](#)

Peran IAM API Gateway

[Peran IAM](#) adalah entitas dalam AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensi sementara dengan API Gateway

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensial keamanan sementara dengan memanggil operasi AWS STS API seperti [AssumeRole](#) atau [GetFederationToken](#)

API Gateway mendukung penggunaan kredensi sementara.

Peran terkait layanan

[Peran terkait AWS layanan](#) memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

API Gateway mendukung peran terkait layanan. Untuk informasi tentang membuat atau mengelola peran terkait layanan API Gateway, lihat. [Menggunakan peran terkait layanan untuk API Gateway](#)

Peran layanan

Layanan dapat mengambil [peran layanan](#) atas nama Anda. Peran layanan memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun, sehingga administrator dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

API Gateway mendukung peran layanan.

Contoh kebijakan berbasis identitas Amazon API Gateway

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya API Gateway. Mereka juga tidak dapat melakukan tugas menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS SDKs. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk informasi tentang cara membuat kebijakan IAM, lihat [Membuat Kebijakan pada Tab JSON di Panduan](#) Pengguna IAM. Untuk informasi tentang tindakan, sumber daya, dan kondisi khusus untuk API Gateway, lihat [Kunci tindakan, sumber daya, dan kondisi untuk Manajemen dan Tindakan Amazon API Gateway, sumber daya, dan kunci kondisi untuk Amazon API Gateway Management V2](#).

Topik

- [Praktik terbaik kebijakan](#)
- [Mengizinkan pengguna melihat izin mereka sendiri](#)
- [Izin baca sederhana](#)
- [Buat hanya REQUEST atau otorisasi JWT](#)
- [Mengharuskan execute-api titik akhir default dinonaktifkan](#)
- [Izinkan pengguna untuk membuat atau memperbarui hanya REST pribadi APIs](#)
- [Mengharuskan rute API memiliki otorisasi](#)
- [Mencegah pengguna membuat atau memperbarui tautan VPC](#)
- [Contoh kebijakan untuk menggunakan aturan perutean](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya API Gateway di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan

yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.

- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat [Validasi kebijakan dengan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Amankan akses API dengan MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan di IAM](#) dalam Panduan Pengguna IAM.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini

mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "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": "*"
    }
  ]
}
```

Izin baca sederhana

Kebijakan contoh ini memberikan izin kepada pengguna untuk mendapatkan informasi tentang semua sumber daya HTTP atau WebSocket API dengan pengenalan a123456789 di AWS Wilayah us-east-1. Sumber daya `arn:aws:apigateway:us-east-1::apis/a123456789/*` mencakup semua sub-sumber daya API seperti otorisasi dan penerapan.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}
```

Buat hanya REQUEST atau otorisasi JWT

Kebijakan contoh ini memungkinkan pengguna untuk membuat APIs dengan hanya REQUEST atau JWT otorisasi, termasuk melalui [impor](#). Di Resource bagian kebijakan, `arn:aws:apigateway:us-east-1::/apis/??????????` sumber daya harus memiliki maksimal 10 karakter, yang mengecualikan sub-sumber daya API. Contoh ini digunakan `ForAllValues` di `Condition` bagian karena pengguna dapat membuat beberapa otorisasi sekaligus dengan mengimpor API.

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"
      ]
    }
  }
}
]
}

```

Mengharuskan **execute-api** titik akhir default dinonaktifkan

Kebijakan contoh ini memungkinkan pengguna untuk membuat, memperbarui, atau mengimpor API, dengan persyaratan yang `DisableExecuteApiEndpoint` adalah `true`. `DisableExecuteApiEndpoint` adalah `true`, klien tidak dapat menggunakan `execute-api` titik akhir default untuk menjalankan API.

Kami menggunakan `BoolIfExists` kondisi untuk menangani panggilan untuk memperbarui API yang tidak memiliki kunci `DisableExecuteApiEndpoint` kondisi terisi. Saat pengguna mencoba membuat atau mengimpor API, kunci `DisableExecuteApiEndpoint` kondisi selalu diisi.

Karena `apis/*` sumber daya juga menangkap sub sumber daya seperti otorisasi atau metode, kami secara eksplisit mencakupnya hanya APIs dengan pernyataan. 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/*/.*"
    ]
}
]
}

```

Izinkan pengguna untuk membuat atau memperbarui hanya REST pribadi APIs

Kebijakan contoh ini menggunakan kunci kondisi untuk mengharuskan pengguna membuat saja PRIVATE APIs, dan untuk mencegah pembaruan yang mungkin mengubah API dari PRIVATE tipe lain, seperti REGIONAL.

Kami menggunakan `ForAllValues` untuk mewajibkan bahwa setiap `EndpointType` ditambahkan ke API adalah PRIVATE. Kami menggunakan kunci kondisi sumber daya untuk mengizinkan pembaruan apa pun ke API selama itu PRIVATE. `ForAllValues` hanya berlaku jika ada kunci kondisi.

Kami menggunakan non-greedy matcher (?) untuk secara eksplisit mencocokkan dengan API IDs untuk mencegah mengizinkan sumber daya non-API seperti otorisasi.

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/*"
        ]
    }
}

```

Mengharuskan rute API memiliki otorisasi

Kebijakan ini menyebabkan upaya untuk membuat atau memperbarui rute (termasuk melalui [impor](#)) gagal jika rute tidak memiliki otorisasi. `ForAnyValue` mengevaluasi ke `false` jika kunci tidak ada, seperti ketika rute tidak sedang dibuat atau diperbarui. Kami menggunakan `ForAnyValue` karena beberapa rute dapat dibuat melalui impor.

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"
        }
    }
}
]
}

```

Mencegah pengguna membuat atau memperbarui tautan VPC

Kebijakan ini mencegah pengguna membuat atau memperbarui tautan VPC. Tautan VPC memungkinkan Anda mengekspos sumber daya dalam VPC Amazon ke klien di luar VPC.

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/*"
      ]
    }
  ]
}

```

```
]
}
```

Contoh kebijakan untuk menggunakan aturan perutean

Contoh kebijakan berikut menunjukkan cara menggunakan tombol `RoutingRule` kondisi untuk mengontrol cara pengguna dapat merutekan lalu lintas dari nama domain kustom mereka ke REST mereka APIs. Anda dapat menggunakan contoh ini untuk membuat kebijakan berbutir halus untuk jenis aturan perutean yang dapat dibuat pengguna. Untuk informasi selengkapnya, lihat [the section called “Aturan perutean”](#).

Mencegah pengguna mengubah cara nama domain kustom merutekan permintaan

Kebijakan ini mencegah pengguna membuat atau memperbarui `BasePathMapping`, `ApiMapping` atau `RoutingRule`. Semua sumber daya ini dapat mengubah cara nama domain kustom merutekan permintaan 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/*"
      ]
    }
  ]
}
```

Memungkinkan pengguna untuk memperbarui aturan routing untuk prioritas tertentu

Kebijakan ini memungkinkan pengguna untuk hanya memperbarui aturan perutean ke prioritas antara 1001 dan 2000. Anda dapat menggunakan aturan ini untuk memisahkan aturan produksi Anda dari aturan prioritas yang lebih rendah dan kemudian mengizinkan pengguna untuk mengubah aturan prioritas yang lebih rendah tanpa memengaruhi aturan produksi.

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
        }
      }
    }
  ]
}
```

Izinkan pengguna memperbarui aturan perutean atau pemetaan jalur dasar untuk nilai jalur dasar tertentu

Kebijakan ini memungkinkan pengguna untuk hanya memperbarui pemetaan jalur dasar untuk setiap jalur dasar yang dimulai dengan `orders` atau memperbarui aturan perutean yang cocok dengan jalur dasar yang dimulai. `orders` Dalam kebijakan ini, pengguna dapat memperbarui aturan pemetaan atau perutean jalur dasar untuk `orders/create` atau `orders123`, tetapi tidak. `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"
        }
      }
    }
  ]
}
```

Izinkan pengguna memperbarui mode perutean ke nilai tertentu

Kebijakan ini memungkinkan pengguna untuk hanya memperbarui mode perutean ke API_MAPPING_ONLY dan ROUTING_RULE_THEN_API_MAPPING. Untuk informasi selengkapnya tentang mode perutean, lihat [the section called “Mengatur mode routing untuk nama domain kustom Anda”](#).

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*"
    }
  }
}
```

Contoh kebijakan berbasis sumber daya Amazon API Gateway

Untuk contoh kebijakan berbasis sumber daya, lihat. [the section called “Contoh kebijakan sumber daya API Gateway”](#)

Memecahkan masalah identitas dan akses Amazon API Gateway

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan API Gateway dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di API Gateway](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya API Gateway saya](#)

Saya tidak berwenang untuk melakukan tindakan di API Gateway

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya `my-example-widget` rekaan, tetapi tidak memiliki izin `apigateway:GetWidget` rekaan.

```
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
```

Dalam hal ini, kebijakan untuk pengguna mateojackson harus diperbarui untuk mengizinkan akses ke sumber daya *my-example-widget* dengan menggunakan tindakan apigateway: *GetWidget*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan iam:PassRole tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke API Gateway.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama marymajor mencoba menggunakan konsol untuk melakukan tindakan di API Gateway. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:  
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan iam:PassRole tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya API Gateway saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis

sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah API Gateway mendukung fitur-fitur ini, lihat [Cara kerja Amazon API Gateway dengan IAM](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Menggunakan peran terkait layanan untuk API Gateway

Amazon API Gateway menggunakan AWS Identity and Access Management peran [terkait layanan](#) (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke API Gateway. Peran terkait layanan telah ditentukan sebelumnya oleh API Gateway dan menyertakan semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan API Gateway lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. API Gateway mendefinisikan izin dari peran terkait layanan, dan kecuali ditentukan lain, hanya API Gateway yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya API Gateway Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat [Layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait

Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk API Gateway

API Gateway menggunakan peran terkait layanan bernama `AWSServiceRoleForAPIGateway`—Memungkinkan API Gateway mengakses Elastic Load Balancing, Amazon Data Firehose, dan sumber daya layanan lainnya atas nama Anda.

Peran `AWSServiceRoleForAPIGateway` terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `ops.apigateway.amazonaws.com`

Kebijakan izin peran memungkinkan API Gateway menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

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/*"
}
]
}

```

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, silakan lihat [Izin Peran Tertaut Layanan](#) di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk API Gateway

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat API, nama domain kustom, atau tautan VPC di, API Gateway Konsol Manajemen AWS AWS CLI, atau AWS API, API Gateway membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat API,

nama domain kustom, atau tautan VPC, API Gateway membuat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk API Gateway

API Gateway tidak memungkinkan Anda mengedit peran `AWSServiceRoleForAPIGateway` terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit Peran Tertaut Layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk API Gateway

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran terkait layanan sebelum menghapusnya secara manual.

Note

Jika layanan API Gateway menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya API Gateway yang digunakan oleh `AWSServiceRoleForAPIGateway`

1. Buka konsol API Gateway di <https://console.aws.amazon.com/apigateway/>.
2. Arahkan ke API, nama domain kustom, atau tautan VPC yang menggunakan peran terkait layanan.
3. Gunakan konsol untuk menghapus sumber daya.
4. Ulangi prosedur untuk menghapus semua APIs, nama domain khusus, atau tautan VPC yang menggunakan peran terkait layanan.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran `AWSServiceRoleForAPIGateway` terkait layanan. Untuk informasi selengkapnya, silakan lihat [Menghapus Peran Terkait Layanan](#) di Panduan Pengguna IAM.

Wilayah yang Didukung untuk peran terkait layanan API Gateway

API Gateway mendukung penggunaan peran terkait layanan di semua Wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [Titik Akhir AWS Layanan](#).

Pembaruan API Gateway ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk API Gateway sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman [riwayat Dokumen](#) API Gateway.

Ubah	Deskripsi	Date
Menambahkan <code>acm:GetCertificate</code> dukungan pada <code>AWSServiceRoleForAPIGateway</code> kebijakan.	<code>AWSServiceRoleForAPIGateway</code> Kebijakan ini sekarang menyertakan izin untuk memanggil tindakan <code>ACM GetCertificate</code> API.	12 Juli 2021
API Gateway mulai melacak perubahan	API Gateway mulai melacak perubahan untuk kebijakan AWS terkelolanya.	12 Juli 2021

Pencatatan dan pemantauan di Amazon API Gateway

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja API Gateway dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. AWS menyediakan beberapa alat untuk memantau sumber daya API Gateway Anda dan menanggapi potensi insiden:

CloudWatch Log Amazon

Untuk membantu men-debug masalah yang terkait dengan eksekusi permintaan atau akses klien ke API Anda, Anda dapat mengaktifkan CloudWatch Log untuk mencatat panggilan API. Untuk informasi selengkapnya, lihat [the section called “CloudWatch log”](#).

CloudWatch Alarm Amazon

Menggunakan CloudWatch alarm, Anda menonton satu metrik selama periode waktu yang Anda tentukan. Jika metrik melebihi ambang batas tertentu, pemberitahuan akan dikirim ke topik atau AWS Auto Scaling kebijakan Amazon Simple Notification Service. CloudWatch alarm tidak memanggil tindakan ketika metrik berada dalam keadaan tertentu. Sebaliknya, kondisi tersebut harus diubah dan dipertahankan selama periode tertentu. Untuk informasi selengkapnya, lihat [the section called “CloudWatch metrik”](#).

Akses Logging ke Firehose

Untuk membantu men-debug masalah yang terkait dengan akses klien ke API Anda, Anda dapat mengaktifkan Firehose untuk mencatat panggilan API. Untuk informasi selengkapnya, lihat [the section called “Firehose”](#).

AWS CloudTrail

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di API Gateway. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat [the section called “Bekerja dengan CloudTrail”](#).

AWS X-Ray

X-Ray adalah AWS layanan yang mengumpulkan data tentang permintaan yang dilayani aplikasi Anda, dan menggunakannya untuk membuat peta layanan yang dapat Anda gunakan untuk mengidentifikasi masalah dengan aplikasi Anda dan peluang untuk pengoptimalan. Untuk informasi selengkapnya, lihat [the section called “Mengatur AWS X-Ray”](#).

AWS Config

AWS Config memberikan tampilan terperinci tentang konfigurasi AWS sumber daya di akun Anda. Anda dapat melihat bagaimana sumber daya terkait, mendapatkan riwayat perubahan konfigurasi, dan melihat bagaimana hubungan dan konfigurasi berubah seiring waktu. Anda dapat menggunakan AWS Config untuk menentukan aturan yang mengevaluasi konfigurasi sumber

daya untuk kepatuhan data. AWS Config aturan mewakili setelan konfigurasi ideal untuk sumber daya API Gateway Anda. Jika sumber daya melanggar aturan dan ditandai sebagai tidak sesuai, AWS Config Anda dapat memperingatkan Anda menggunakan topik Amazon Simple Notification Service (Amazon SNS). Untuk detailnya, lihat [the section called “Bekerja dengan AWS Config”](#).

Mencatat panggilan API Amazon API Gateway menggunakan AWS CloudTrail

Amazon API Gateway terintegrasi dengan [AWS CloudTrail](#), layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan REST API untuk layanan API Gateway sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol API Gateway dan panggilan kode ke layanan API Gateway APIs. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP dari mana permintaan dibuat, kapan dibuat, dan detail tambahan.

Note

[TestInvokeAuthorizer](#) dan [TestInvokeMethod](#) tidak masuk CloudTrail.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. Wilayah AWS Untuk informasi selengkapnya, lihat [Bekerja dengan riwayat CloudTrail Acara](#) di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara [CloudTrailDanau](#).

CloudTrail jalan setapak

Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan Konsol Manajemen AWS Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan. AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. Wilayah AWS Untuk informasi selengkapnya tentang jejak, lihat [Membuat jejak untuk Anda Akun AWS](#) dan [Membuat jejak untuk organisasi](#) di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#). Untuk informasi tentang harga Amazon S3, lihat [Harga Amazon S3](#).

CloudTrail Menyimpan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail [Lake mengonversi peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC](#). ORC adalah format penyimpanan kolom yang dioptimalkan untuk pengambilan data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara [tingkat lanjut](#). Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat [Bekerja dengan AWS CloudTrail Danau](#) di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih [opsi harga](#) yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa, dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#).

Acara manajemen API Gateway di CloudTrail

[Acara manajemen](#) memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di Anda Akun AWS. Ini juga dikenal sebagai operasi bidang kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

Amazon API Gateway mencatat semua tindakan API Gateway sebagai peristiwa manajemen, kecuali untuk [TestInvokeAuthorizer](#) dan [TestInvokeMethod](#). Untuk mengetahui daftar tindakan Amazon API Gateway yang dicatatkan oleh API Gateway CloudTrail, lihat [Referensi API Amazon API Gateway](#).

Contoh acara API Gateway

Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang operasi API yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga peristiwa tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail peristiwa yang menunjukkan GetResource tindakan API Gateway:

```
{
  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 ...
]
```

Untuk informasi tentang konten CloudTrail rekaman, lihat [konten CloudTrail rekaman](#) di Panduan AWS CloudTrail Pengguna.

Memantau konfigurasi API Gateway API dengan AWS Config

Anda dapat menggunakan [AWS Config](#) untuk merekam perubahan konfigurasi yang dibuat pada resource API Gateway API dan mengirim notifikasi berdasarkan perubahan sumber daya. Mempertahankan riwayat perubahan konfigurasi untuk sumber daya API Gateway berguna untuk pemecahan masalah operasional, audit, dan kasus penggunaan kepatuhan.

AWS Config dapat melacak perubahan ke:

- Konfigurasi tahap API, seperti:
 - pengaturan cluster cache
 - pengaturan throttle
 - pengaturan log akses
 - penyebaran aktif yang ditetapkan di atas panggung
- Konfigurasi API, seperti:
 - konfigurasi titik akhir
 - versi
 - protokol
 - tag

Selain itu, Aturan AWS Config fitur ini memungkinkan Anda untuk menentukan aturan konfigurasi dan secara otomatis mendeteksi, melacak, dan memperingatkan pelanggaran terhadap aturan ini. Dengan melacak perubahan pada properti konfigurasi sumber daya ini, Anda juga dapat membuat AWS Config aturan yang dipicu perubahan untuk sumber daya API Gateway Anda, dan menguji konfigurasi sumber daya Anda terhadap praktik terbaik.

Anda dapat mengaktifkan AWS Config di akun Anda dengan menggunakan AWS Config konsol atau AWS CLI. Pilih jenis sumber daya yang ingin Anda lacak perubahannya. Jika sebelumnya Anda mengonfigurasi AWS Config untuk merekam semua jenis sumber daya, sumber daya API Gateway ini akan direkam secara otomatis di akun Anda. Support untuk Amazon API Gateway in AWS Config tersedia di semua wilayah AWS publik dan AWS GovCloud (US). Untuk daftar lengkap Wilayah yang didukung, lihat [Titik Akhir dan Kuota Amazon API Gateway](#) di bagian. Referensi Umum AWS

Topik

- [Jenis sumber daya yang mendukung](#)
- [Menyiapkan AWS Config](#)
- [Mengkonfigurasi AWS Config untuk merekam sumber daya API Gateway](#)
- [Melihat detail konfigurasi API Gateway di AWS Config konsol](#)
- [Mengevaluasi sumber daya API Gateway menggunakan aturan AWS Config](#)

Jenis sumber daya yang mendukung

Jenis sumber daya API Gateway berikut terintegrasi dengan AWS Config dan didokumentasikan dalam [Jenis AWS Sumber Daya dan Hubungan Sumber Daya yang AWS Config Didukung](#):

- `AWS::ApiGatewayV2::Api`(WebSocket dan HTTP API)
- `AWS::ApiGateway::RestApi`(REST API)
- `AWS::ApiGatewayV2::Stage`(WebSocket dan tahap HTTP API)
- `AWS::ApiGateway::Stage`(Tahap REST API)

Untuk informasi selengkapnya AWS Config, lihat [Panduan AWS Config Pengembang](#). Untuk informasi harga, lihat [halaman informasi harga AWS Config](#).

Important

Jika Anda mengubah salah satu properti API berikut setelah API diterapkan, Anda harus menerapkan [ulang API untuk menyebarkan](#) perubahan. Jika tidak, Anda akan melihat perubahan atribut di AWS Config konsol, tetapi pengaturan properti sebelumnya akan tetap berlaku; perilaku runtime API tidak akan berubah.

- **AWS::ApiGateway::RestApi** – binaryMediaTypes, minimumCompressionSize, apiKeySource
- **AWS::ApiGatewayV2::Api** – apiKeySelectionExpression

Menyiapkan AWS Config

Untuk mengatur awalnya AWS Config, lihat topik berikut di [Panduan AWS Config Pengembang](#).

- [Menyiapkan AWS Config dengan Konsol](#)
- [Menyiapkan AWS Config dengan AWS CLI](#)

Mengkonfigurasi AWS Config untuk merekam sumber daya API Gateway

Secara default, AWS Config mencatat perubahan konfigurasi untuk semua jenis sumber daya regional yang didukung yang ditemukan di wilayah tempat lingkungan Anda berjalan. Anda dapat menyesuaikan AWS Config untuk merekam perubahan hanya untuk jenis sumber daya tertentu, atau perubahan pada sumber daya global.

Untuk mempelajari sumber daya regional vs global dan mempelajari cara menyesuaikan AWS Config konfigurasi Anda, lihat [Memilih AWS Config Rekaman Sumber Daya yang mana](#).

Melihat detail konfigurasi API Gateway di AWS Config konsol

Anda dapat menggunakan AWS Config konsol untuk mencari sumber daya API Gateway dan mendapatkan detail terkini dan historis tentang konfigurasinya. Prosedur berikut menunjukkan cara menemukan informasi tentang API Gateway API.

Untuk menemukan sumber daya API Gateway di AWS konsol konfigurasi

1. Buka [konsol AWS Config](#).
2. Pilih Sumber Daya.

3. Pada halaman Inventarisasi sumber daya, pilih Sumber Daya.
4. Buka menu Jenis sumber daya, gulir ke API Gateway atau API Gateway V2, lalu pilih satu atau beberapa jenis sumber daya API Gateway.
5. Pilih Lihat.
6. Pilih ID sumber daya dalam daftar sumber daya yang AWS Config ditampilkan. AWS Config menampilkan detail konfigurasi dan informasi lain tentang sumber daya yang Anda pilih.
7. Untuk melihat detail lengkap konfigurasi yang direkam, pilih Lihat Detail.

Untuk mempelajari lebih lanjut cara menemukan sumber daya dan melihat informasi di halaman ini, lihat [Melihat Konfigurasi dan Riwayat AWS Sumber Daya](#) di Panduan AWS Config Pengembang.

Mengevaluasi sumber daya API Gateway menggunakan aturan AWS Config

Anda dapat membuat AWS Config aturan, yang mewakili pengaturan konfigurasi ideal untuk sumber daya API Gateway Anda. Anda dapat menggunakan [Aturan AWS Config Terkelola](#) yang telah ditentukan sebelumnya, atau menentukan aturan khusus. AWS Config terus melacak perubahan pada konfigurasi sumber daya Anda untuk menentukan apakah perubahan tersebut melanggar salah satu kondisi dalam aturan Anda. AWS Config Konsol menunjukkan status kepatuhan aturan dan sumber daya Anda.

Jika sumber daya melanggar aturan dan ditandai sebagai tidak patuh, AWS Config dapat mengingatkan Anda menggunakan topik Amazon [Simple Notification Service Developer Guide](#) (Amazon SNS). Untuk menggunakan data dalam AWS Config peringatan ini secara terprogram, gunakan antrian Amazon Simple Queue Service (Amazon SQS) sebagai titik akhir notifikasi untuk topik Amazon SNS.

Untuk mempelajari lebih lanjut tentang menyiapkan dan menggunakan aturan, lihat [Mengevaluasi Sumber Daya dengan Aturan](#) di [Panduan AWS Config Pengembang](#).

Validasi kepatuhan untuk Amazon API Gateway

Untuk mempelajari apakah Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#).

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#).

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat [Dokumentasi AWS Keamanan](#).

Ketahanan di Amazon API Gateway

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Sebagai layanan Regional yang dikelola sepenuhnya, API Gateway beroperasi di beberapa Availability Zone di setiap Wilayah, menggunakan redundansi Availability Zone untuk meminimalkan kegagalan infrastruktur sebagai kategori risiko ketersediaan. API Gateway dirancang untuk memulihkan secara otomatis dari kegagalan Availability Zone.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [Infrastruktur AWS Global](#).

Untuk APIs mencegah Anda kewalahan oleh terlalu banyak permintaan, API Gateway membatasi permintaan ke permintaan Anda. APIs Secara khusus, API Gateway menetapkan batas pada tingkat kondisi tunak dan ledakan pengiriman permintaan terhadap semua APIs yang ada di akun Anda. Anda dapat mengonfigurasi pelambatan khusus untuk Anda. APIs Untuk mempelajari selengkapnya, lihat [Throttle request ke REST APIs Anda untuk throughput yang lebih baik di API Gateway](#).

Anda dapat menggunakan pemeriksaan kesehatan Route 53 untuk mengontrol failover DNS dari API Gateway API di wilayah utama ke API Gateway API di wilayah sekunder. Sebagai contoh, lihat [the section called “failover DNS”](#).

Keamanan infrastruktur di Amazon API Gateway

Sebagai layanan terkelola, Amazon API Gateway dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik

untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses API Gateway melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun, tetapi API Gateway mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan berbasis sumber daya untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau spesifik. VPCs Secara efektif, ini mengisolasi akses jaringan ke sumber daya API Gateway tertentu hanya dari VPC tertentu dalam AWS jaringan.

Analisis kerentanan di Amazon API Gateway

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat [model tanggung jawab AWS bersama](#).

Praktik terbaik keamanan di Amazon API Gateway

API Gateway menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Terapkan akses hak akses paling rendah

Gunakan kebijakan IAM untuk menerapkan akses hak istimewa paling sedikit untuk membuat, membaca, memperbarui, atau menghapus API Gateway. APIs Untuk mempelajari selengkapnya, lihat [Manajemen identitas dan akses untuk Amazon API Gateway](#). API Gateway menawarkan beberapa opsi untuk mengontrol akses APIs yang Anda buat. Untuk mempelajari lebih lanjut,

lihat [Kontrol dan kelola akses ke REST APIs di API Gateway](#), [Mengontrol dan mengelola akses ke WebSocket APIs dalam API Gateway](#), dan [Kontrol akses ke HTTP APIs dengan otorisasi JWT di API Gateway](#).

Menerapkan logging

Gunakan CloudWatch Log atau Amazon Data Firehose untuk mencatat permintaan ke Anda. APIs Untuk mempelajari lebih lanjut, lihat [Pantau REST APIs di API Gateway](#), [Konfigurasi logging untuk WebSocket APIs di API Gateway](#), dan [Konfigurasi logging untuk HTTP APIs di API Gateway](#).

Menerapkan CloudWatch alarm Amazon

Menggunakan CloudWatch alarm, Anda menonton satu metrik selama periode waktu yang Anda tentukan. Jika metrik melebihi ambang batas tertentu, pemberitahuan akan dikirim ke topik atau AWS Auto Scaling kebijakan Amazon Simple Notification Service. CloudWatch alarm tidak memanggil tindakan ketika metrik berada dalam keadaan tertentu. Sebaliknya, negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat [the section called “CloudWatch metrik”](#).

Aktifkan AWS CloudTrail

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di API Gateway. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat [the section called “Bekerja dengan CloudTrail”](#).

Aktifkan AWS Config

AWS Config memberikan tampilan terperinci tentang konfigurasi AWS sumber daya di akun Anda. Anda dapat melihat bagaimana sumber daya terkait, mendapatkan riwayat perubahan konfigurasi, dan melihat bagaimana hubungan dan konfigurasi berubah seiring waktu. Anda dapat menggunakan AWS Config untuk menentukan aturan yang mengevaluasi konfigurasi sumber daya untuk kepatuhan data. AWS Config aturan mewakili pengaturan konfigurasi ideal untuk sumber daya API Gateway Anda. Jika sumber daya melanggar aturan dan ditandai sebagai tidak sesuai, AWS Config dapat mengingatkan Anda menggunakan topik Amazon Simple Notification Service (Amazon SNS). Lihat perinciannya di [the section called “Bekerja dengan AWS Config”](#).

Gunakan AWS Security Hub CSPM

Pantau penggunaan API Gateway yang berkaitan dengan praktik terbaik keamanan dengan menggunakan [AWS Security Hub CSPM](#). Security Hub CSPM menggunakan kontrol keamanan

untuk mengevaluasi konfigurasi sumber daya dan standar keamanan untuk membantu Anda mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya tentang menggunakan Security Hub CSPM guna mengevaluasi resource API Gateway, lihat [kontrol Amazon API Gateway](#) di AWS Security Hub Panduan Pengguna.

Menandai sumber daya API Gateway Anda

Tag adalah label metadata yang Anda tetapkan atau yang ditetapkan ke sumber AWS daya. AWS Setiap tag memiliki dua bagian:

- Sebuah kunci tag (misalnya, CostCenter, Environment, atau Project). Kunci tag peka huruf besar dan kecil.
- Bidang opsional yang dikenal sebagai nilai tag (misalnya, 111122223333 atau Production). Mengabaikan nilai tag sama dengan menggunakan sebuah string kosong. Seperti kunci tag, nilai tag peka huruf besar/kecil.

Tanda membantu Anda melakukan hal berikut:

- Kontrol akses ke sumber daya Anda berdasarkan tanda yang ditetapkan untuk sumber daya tersebut. Anda mengontrol akses dengan menentukan kunci tanda dan nilai-nilai dalam syarat untuk (IAM) kebijakan AWS Identity and Access Management . Untuk informasi selengkapnya tentang kontrol akses berbasis tag, lihat [Mengontrol Akses Menggunakan Tag](#) di Panduan Pengguna IAM.
- Lacak AWS biaya Anda. Anda mengaktifkan tag ini di AWS Manajemen Penagihan dan Biaya dasbor. AWS menggunakan tag untuk mengkategorikan biaya Anda dan mengirimkan laporan alokasi biaya bulanan kepada Anda. Untuk informasi selengkapnya, lihat [Menggunakan Tag Alokasi Biaya](#) dalam [Panduan Pengguna AWS Billing](#).
- Identifikasi dan atur AWS sumber daya Anda. Banyak AWS layanan mendukung penandaan, sehingga Anda dapat menetapkan tag yang sama ke sumber daya dari layanan yang berbeda untuk menunjukkan bahwa sumber daya terkait. Misalnya, Anda dapat menetapkan tag yang sama ke tahap API Gateway yang Anda tetapkan ke aturan CloudWatch Peristiwa.

Untuk tips menggunakan tag, lihat whitepaper Strategi [AWS Penandaan](#).

Bagian berikut memberikan informasi selengkapnya tentang tag untuk Amazon API Gateway.

Topik

- [Sumber daya API Gateway yang dapat ditandai](#)
- [Menggunakan tag untuk mengontrol akses ke sumber daya API API Gateway REST API](#)

Sumber daya API Gateway yang dapat ditandai

Tag dapat disetel pada API HTTP atau sumber daya WebSocket API berikut di [Amazon API Gateway V2 API](#):

- Api
- DomainName
- Stage
- VpcLink

Selain itu, tag dapat disetel pada resource REST API berikut di [Amazon API Gateway V1 API](#):

- ApiKey
- ClientCertificate
- DomainName
- DomainNameAccessAssociation
- RestApi
- Stage
- UsagePlan
- VpcLink

Tag tidak dapat disetel langsung pada sumber daya lain. Namun, di [Amazon API Gateway V1 API](#), sumber daya turunan mewarisi tag yang disetel pada sumber daya induk. Contoh:

- Jika tag disetel pada RestApi sumber daya, tag tersebut diwarisi oleh sumber daya turunan berikut RestApi untuk kontrol akses [berbasis Atribut](#):
 - Authorizer
 - Deployment
 - Documentation
 - GatewayResponse
 - Integration
 - Method
 - Model

- Resource
 - ResourcePolicy
 - Setting
 - Stage
- Jika tag disetel pada `DomainName`, tag tersebut diwarisi oleh salah satu `BasePathMappingApiMapping`, dan `RoutingRule` sumber daya di bawahnya.
 - Jika tag disetel pada `UsagePlan`, tag tersebut diwarisi oleh `UsagePlanKey` sumber daya apa pun di bawahnya.

Note

Warisan tag hanya berlaku untuk kontrol [akses berbasis atribut](#). Misalnya, Anda tidak dapat menggunakan tag yang diwariskan untuk memantau biaya masuk AWS Cost Explorer. Untuk menggunakan tag untuk alokasi biaya, sebaiknya Anda membuat tag pada sumber daya anak, seperti Stage sumber daya.

API Gateway tidak menampilkan tag yang diwariskan saat Anda memanggil [GetTags](#) sumber daya.

Warisan tag di Amazon API Gateway V1

Sebelumnya hanya mungkin untuk mengatur tag pada tahapan. Sekarang Anda juga dapat mengaturnya pada sumber daya lain, a Stage dapat menerima tag dua cara:

- Tag dapat diatur langsung pada `fileStage`.
- Tahap dapat mewarisi tag dari `RestApi` induknya.

Jika sebuah stage menerima tag dua arah, tag yang disetel langsung di atas panggung akan diutamakan. Misalnya, sebuah stage mewarisi tag berikut dari REST API induknya:

```
{
  'foo': 'bar',
  'x': 'y'
}
```

Misalkan juga memiliki tag berikut yang disetel secara langsung:

```
{
  'foo': 'bar2',
  'hello': 'world'
}
```

Efek bersihnya adalah untuk tahap memiliki tag berikut, dengan nilai-nilai berikut:

```
{
  'foo': 'bar2',
  'hello': 'world'
  'x': 'y'
}
```

Pembatasan tag dan konvensi penggunaan

Pembatasan dan konvensi penggunaan berikut berlaku untuk penggunaan tag dengan sumber daya API Gateway:

- Setiap sumber daya dapat memiliki maksimum 50 tag.
- Untuk setiap sumber daya, setiap kunci tag harus unik, dan setiap kunci tag hanya dapat memiliki satu nilai.
- Panjang kunci tag maksimum adalah 128 karakter Unicode dalam UTF-8.
- Panjang nilai tag maksimum adalah 256 karakter Unicode dalam UTF-8.
- Karakter yang diizinkan untuk kunci dan nilai adalah huruf, angka, spasi yang dapat direpresentasikan dalam UTF-8, dan karakter berikut: `.:+=@_/-` (tanda hubung). Sumber daya Amazon EC2 memungkinkan karakter apa pun.
- Kunci dan nilai tag peka terhadap huruf besar dan kecil. Sebagai praktik terbaik, putuskan strategi untuk memanfaatkan tag dan terapkan strategi tersebut secara konsisten di semua jenis sumber daya. Misalnya, putuskan apakah akan menggunakan `Costcenter`, `costcenter`, atau `CostCenter` dan menggunakan kesepakatan yang sama untuk semua tag. Hindari penggunaan tag yang serupa dengan perlakuan kasus yang tidak konsisten.
- `aws:` Awalan dilarang untuk tag; itu dicadangkan untuk AWS digunakan. Anda tidak dapat mengedit atau menghapus kunci atau nilai tanda dengan prefiks ini. Tanda dengan prefiks ini tidak dihitung, berlawanan dengan tanda milik Anda per batas sumber daya.

Menggunakan tag untuk mengontrol akses ke sumber daya API

API Gateway REST API

Ketentuan dalam AWS Identity and Access Management kebijakan adalah bagian dari sintaks yang Anda gunakan untuk menentukan izin ke sumber daya API Gateway. Untuk detail tentang menentukan kebijakan IAM, lihat [the section called “Gunakan izin IAM”](#) Di API Gateway, sumber daya dapat memiliki tag, dan beberapa tindakan dapat menyertakan tag. Saat membuat kebijakan IAM, Anda dapat menggunakan kunci kondisi tag untuk mengontrol:

- Pengguna mana yang dapat melakukan tindakan pada sumber daya API Gateway, berdasarkan tag yang sudah dimiliki sumber daya.
- Tanda apa yang dapat diteruskan dalam permintaan tindakan.
- Apakah kunci tanda tertentu dapat digunakan dalam permintaan.

Menggunakan tag untuk kontrol akses berbasis atribut dapat memungkinkan kontrol yang lebih baik daripada kontrol tingkat API, serta kontrol yang lebih dinamis daripada kontrol akses berbasis sumber daya. Kebijakan IAM dapat dibuat yang mengizinkan atau melarang operasi berdasarkan tag yang disediakan dalam permintaan (tag permintaan), atau tag pada sumber daya yang sedang dioperasikan (tag sumber daya). Secara umum, tag sumber daya adalah untuk sumber daya yang sudah ada. Tag permintaan digunakan saat Anda membuat sumber daya baru.

Untuk sintaks dan semantik kunci syarat tanda yang lengkap, lihat [Controlling Access Using Tags](#) dalam Panduan Pengguna IAM.

Contoh berikut menunjukkan cara menentukan kondisi tag dalam kebijakan untuk pengguna API Gateway.

Batasi tindakan berdasarkan tag sumber daya

Contoh kebijakan berikut memberi pengguna izin untuk melakukan semua tindakan pada semua sumber daya, selama sumber daya tersebut tidak memiliki tag `Environment` dengan nilai `prod`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

    {
      "Effect": "Allow",
      "Action": "apigateway:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}

```

Izinkan tindakan berdasarkan tag sumber daya

Contoh kebijakan berikut memungkinkan pengguna untuk melakukan semua tindakan pada sumber daya API Gateway, selama sumber daya memiliki tag `Environment` dengan nilai `Development`. DenyPernyataan tersebut mencegah pengguna mengubah nilai `Environment` tag.

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"
      }
    }
  }
]
}

```

Tolak operasi penandaan

Contoh kebijakan berikut memungkinkan pengguna untuk melakukan semua tindakan API Gateway, kecuali untuk mengubah tag.

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "apigateway:*"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Deny",
    "Action": [
      "apigateway:*"
    ],
    "Resource": "arn:aws:apigateway:*::/tags*"
  }
]
}
```

Izinkan operasi penandaan

Contoh kebijakan berikut memungkinkan pengguna untuk mendapatkan semua sumber daya API Gateway, dan mengubah tag untuk sumber daya tersebut. Untuk mendapatkan tag untuk sumber daya, pengguna harus memiliki GET izin untuk sumber daya tersebut. Untuk memperbarui tag untuk sumber daya, pengguna harus memiliki PATCH izin untuk sumber daya tersebut.

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:*::*"
        ]
    }
]
```

Referensi API

Amazon API Gateway menyediakan APIs untuk membuat dan menerapkan HTTP Anda sendiri dan WebSocket APIs. Selain itu, API Gateway APIs tersedia dalam standar AWS SDKs.

Jika Anda menggunakan bahasa yang memiliki AWS SDK, Anda dapat memilih untuk menggunakan SDK daripada menggunakan API Gateway APIs REST secara langsung. SDKs Membuat otentikasi lebih sederhana, terintegrasi dengan mudah dengan lingkungan pengembangan Anda, dan menyediakan akses mudah ke perintah API Gateway.

Di sinilah tempat menemukan dokumentasi referensi API REST API AWS SDKs dan API Gateway:

- [Alat untuk Amazon Web Services](#)
- [Referensi API REST API Amazon API Gateway](#)
- [Amazon API Gateway WebSocket dan Referensi HTTP API](#)

Titik akhir layanan API Gateway

Endpoint adalah URL yang berfungsi sebagai titik masuk untuk layanan AWS web. API Gateway mendukung tipe endpoint berikut:

- [the section called “IPv4 titik akhir”](#)
- [Titik akhir dualstack](#) (mendukung keduanya dan) IPv4 IPv6
- [Titik akhir FIPS](#)

Saat Anda membuat permintaan, Anda dapat menentukan titik akhir yang akan digunakan. Jika Anda tidak menentukan titik akhir, IPv4 titik akhir digunakan secara default. Untuk menggunakan tipe titik akhir yang berbeda, Anda harus menentukannya dalam permintaan Anda. Untuk contoh cara melakukannya, lihat [the section called “Menentukan titik akhir”](#). Untuk tabel titik akhir yang tersedia, lihat titik akhir [Amazon API Gateway](#).

IPv4 titik akhir

IPv4 endpoint hanya mendukung IPv4 lalu lintas. IPv4 titik akhir tersedia untuk semua Wilayah.

Jika Anda menentukan titik akhir umum, `apigateway.amazonaws.com`, kami menggunakan titik akhir untuk `us-east-1`. Untuk menggunakan Wilayah yang berbeda, tentukan titik akhir yang terkait.

Misalnya, jika Anda menentukan `apigateway.us-east-2.amazonaws.com` sebagai titik akhir, kami mengarahkan permintaan Anda ke titik `us-east-2` akhir.

IPv4 nama endpoint menggunakan konvensi penamaan berikut:

- `apigateway.region.amazonaws.com`

Misalnya, nama IPv4 endpoint untuk `eu-west-1` Region adalah `apigateway.eu-west-1.amazonaws.com`.

Titik akhir Dualstack (IPv4 dan IPv6)

Dualstack endpoint mendukung keduanya IPv4 dan lalu lintas IPv6. Saat Anda membuat permintaan ke titik akhir dualstack, URL endpoint akan diselesaikan ke alamat IPv6 atau IPv4 alamat, tergantung pada protokol yang digunakan oleh jaringan dan klien Anda.

Nama titik akhir tumpukan ganda menggunakan konvensi penamaan berikut:

- `apigateway.region.api.aws`

Misalnya, nama titik akhir tumpukan ganda untuk Wilayah `eu-west-1` adalah `apigateway.eu-west-1.api.aws`.

Menentukan titik akhir

Contoh berikut menunjukkan cara menentukan titik akhir untuk `us-east-2` Wilayah menggunakan AWS CLI `forapigateway`.

- Tumpukan ganda

```
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
```

Contoh berikut menunjukkan cara menentukan titik akhir untuk us-east-2 Wilayah menggunakan AWS CLI forapigatewayv2.

- Tumpukan ganda

```
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
```

Menggunakan IPv6 alamat dalam kebijakan IAM

Jika Anda menggunakan kebijakan pengguna IAM atau kebijakan sumber daya API Gateway untuk mengontrol akses ke API Gateway atau API Gateway apa pun APIs, konfirmasi bahwa kebijakan Anda diperbarui untuk menyertakan rentang IPv6 alamat. Kebijakan yang tidak diperbarui untuk menangani IPv6 alamat dapat memengaruhi akses klien ke API Gateway saat mereka mulai menggunakan titik akhir dualstack. Untuk informasi selengkapnya tentang mengelola izin akses dengan IAM, lihat [Manajemen identitas dan akses untuk Amazon API Gateway](#)

Kebijakan IAM yang memfilter alamat IP menggunakan [Operator Kondisi Alamat IP](#). Kebijakan identitas berikut memungkinkan alamat IP dalam 54.240.143.* rentang untuk mendapatkan informasi tentang semua sumber daya HTTP atau WebSocket API dengan a123456789 pengenal. Alamat IP apa pun di luar rentang ini akan ditolak mengakses informasi tentang semua sumber daya di API. Karena semua IPv6 alamat berada di luar rentang yang diizinkan, kebijakan ini mencegah IPv6 alamat agar tidak dapat mengakses informasi tentang API.

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"}
    }
}
]
}

```

Anda dapat memodifikasi Condition elemen kebijakan API untuk mengizinkan rentang alamat IPv4 IPv6 (54.240.143.0/24 2001:DB8:1234:5678::/64) dan () seperti yang ditunjukkan pada contoh berikut. Anda dapat menggunakan jenis Condition blok yang sama yang ditunjukkan dalam contoh untuk memperbarui kebijakan pengguna IAM dan kebijakan sumber daya API Gateway.

```

"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}

```

Kuota Amazon API Gateway

Kuota berikut berlaku untuk semua jenis API Amazon API Gateway.

Kuota tingkat akun API Gateway, per Wilayah

Kuota berikut berlaku per akun, per Wilayah di Amazon API Gateway.

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Throttle kuota per akun, per Wilayah di HTTP, REST APIs APIs, WebSocket APIs dan callback WebSocket APIs	10.000 permintaan per detik (RPS) dengan kapasitas burst tambahan yang disediakan oleh algoritma token bucket , menggunakan kapasitas bucket maksimum 5.000 permintaan. *	Ya
 Note Kuota burst ditentukan oleh tim layanan API Gateway berdasarkan keseluruhan kuota RPS untuk akun di Wilayah. Ini bukan kuota yang dapat dikendalikan atau diminta oleh pelanggan untuk melakukan perubahan.		
Throttle kuota tanpa kontrol akses per akun per Wilayah untuk portal	250.000 permintaan per detik	Tidak
Throttle kuota dengan kontrol akses per akun per Wilayah untuk portal	10.000 permintaan per detik	Tidak

* Untuk Wilayah berikut, kuota throttle default adalah 2500 RPS dan kuota burst default adalah 1250 RPS: Afrika (Cape Town), Eropa (Milan), Asia Pasifik (Jakarta), Timur Tengah (UEA), Asia Pasifik (Hyderabad), Asia Pasifik (Melbourne), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), Kanada Barat (Calgary), Asia Pasifik (Malaysia), Asia Pasifik (Thailand), dan Meksiko (Tengah).

Kuota API Gateway untuk membuat, menerapkan, dan mengelola API

Kuota tetap berikut berlaku untuk membuat, menerapkan, dan mengelola API di API Gateway, menggunakan, konsol API Gateway AWS CLI, atau API Gateway REST API dan nya. SDKs Kuota ini tidak dapat ditingkatkan.

Tindakan	Kuota default	Dapat ditingkatkan
CreateApiKey	5 permintaan per detik per akun	Tidak
CreateDeployment	1 permintaan setiap 5 detik per akun	Tidak
CreateDocumentationVersion	1 permintaan setiap 20 detik per akun	Tidak
CreateDomainName	1 permintaan setiap 30 detik per akun	Tidak
CreateResource	5 permintaan per detik per akun	Tidak
CreateRestApi untuk API Regional atau pribadi	1 permintaan setiap 3 detik per akun	Tidak
CreateRestApi untuk API yang dioptimalkan tepi	1 permintaan setiap 30 detik per akun	Tidak
CreateVpcLink (V2)	1 permintaan setiap 15 detik per akun	Tidak

Tindakan	Kuota default	Dapat ditingkatkan
DeleteApiKey	5 permintaan per detik per akun	Tidak
DeleteDomainName	1 permintaan setiap 30 detik per akun	Tidak
DeleteResource	5 permintaan per detik per akun	Tidak
DeleteRestApi	1 permintaan setiap 30 detik per akun	Tidak
GetResources	5 permintaan setiap 2 detik per akun	Tidak
DeleteVpcLink(V2)	1 permintaan setiap 30 detik per akun	Tidak
ImportDocumentationParts	1 permintaan setiap 30 detik per akun	Tidak
ImportRestApi untuk API Regional atau pribadi	1 permintaan setiap 3 detik per akun	Tidak
ImportRestApi untuk API yang dioptimalkan tepi	1 permintaan setiap 30 detik per akun	Tidak
PutRestApi	1 permintaan per detik per akun	Tidak
UpdateAccount	1 permintaan setiap 20 detik per akun	Tidak
UpdateDomainName	1 permintaan setiap 30 detik per akun	Tidak
UpdateUsagePlan	1 permintaan setiap 20 detik per akun	Tidak

Tindakan	Kuota default	Dapat ditingkatkan
Buat Portal	1 permintaan setiap 3 detik	Tidak
Perbarui Portal	2 permintaan per menit	Tidak
Dapatkan Portal	10 permintaan per detik	Tidak
Daftar Portal	10 permintaan per detik	Tidak
Publikasikan Portal	2 permintaan per menit	Tidak
DeletePortal	2 permintaan per menit	Tidak
PreviewPortal	1 permintaan setiap 3 detik	Tidak
DisablePortal	2 permintaan per menit	Tidak
GetPortalProduct	10 permintaan per detik	Tidak
ListPortalProduct	5 permintaan per detik	Tidak
CreatePortalProduct	2 permintaan per detik	Tidak
UpdatePortalProduct	0,5 permintaan per detik	Tidak
DeletePortalProduct	1 permintaan per detik	Tidak
PutPortalProductSharingPolicy	1 permintaan setiap 3 detik	Tidak
GetPortalProductSharingPolicy	10 permintaan per detik	Tidak
DeletePortalProductSharingPolicy	1 permintaan setiap 3 detik	Tidak
CreateProductRestEndpointPage	1 permintaan setiap 3 detik	Tidak
GetProductRestEndpointPage	10 permintaan per detik	Tidak

Tindakan	Kuota default	Dapat ditingkatkan
UpdateProductRestEndpointPage	1 permintaan setiap 3 detik	Tidak
DeleteProductRestEndpointPage	1 permintaan setiap 3 detik	Tidak
ListProductRestEndpoints	10 permintaan per detik	Tidak
CreateProductPage	1 permintaan setiap 3 detik	Tidak
GetProductPage	10 permintaan per detik	Tidak
UpdateProductPage	1 permintaan setiap 3 detik	Tidak
DeleteProductPage	1 permintaan setiap 3 detik	Tidak
ListProductPages	10 permintaan per detik	Tidak
Operasi lainnya	Tidak ada kuota hingga total kuota akun.	Tidak
Total operasi	10 permintaan per detik dengan kuota burst 40 permintaan per detik.	Tidak

Kuota untuk mengonfigurasi dan menjalankan REST API di API Gateway

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan REST API di Amazon API Gateway. Jika kuota adalah per-API, itu hanya dapat ditingkatkan secara spesifik APIs, tidak untuk semua yang ada APIs di akun.

Untuk informasi tentang kuota tingkat akun, lihat [Kuota dan catatan penting](#)

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Nama domain kustom publik per akun per Wilayah	120	Ya
Nama domain kustom pribadi per akun per Wilayah	50	Ya
Regional APIs per Wilayah	600	Tidak
Tepi dioptimalkan per Wilayah APIs	120	Tidak
Pribadi APIs per akun per Wilayah	600	Tidak
Asosiasi akses nama domain per akun	100	Ya
Pemetaan API multi-level per domain	200	Tidak
Aturan perutean per domain	50	Ya
Panjang, dalam karakter, URL untuk API yang dioptimalkan tepi	8192	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Panjang, dalam karakter, URL untuk API Regional	10240	Tidak
Panjang, dalam karakter, URL untuk API pribadi	8192	Tidak
Panjang, dalam karakter, kebijakan sumber daya API Gateway	8192	Ya
Kunci API per akun per Wilayah	10000	Tidak
Sertifikat klien per akun per Wilayah	60	Ya
Otorisasi per API (AWS Lambda dan Amazon Cognito)	10	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Bagian dokumentasi per API	2000	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center
Sumber daya per API	300	Ya
Stage per API	10	Ya
Variabel stage per stage	100	Tidak
Panjang, dalam karakter, dari kunci dalam variabel tahap	64	Tidak
Panjang, dalam karakter, dari nilai dalam variabel tahap	512	Tidak
Paket penggunaan per akun per Wilayah	300	Ya

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Paket penggunaan per kunci API	10	Ya
Tautan VPC per akun per Wilayah	20	Ya
API caching TTL	300 detik secara default dan dapat dikonfigurasi antara 0 dan 3600 oleh pemilik API.	Bukan untuk batas atas (3600)
Ukuran respons yang di-cache	1048576 Byte. Enkripsi data cache dapat meningkatkan ukuran item yang sedang di-cache.	Tidak
Batas waktu integrasi untuk Regional APIs	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	IYA*
Batas waktu integrasi untuk dioptimalkan tepi APIs	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	Tidak
Batas waktu integrasi untuk pribadi APIs	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	IYA*

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Total ukuran gabungan dari semua nilai header, termasuk nama header, nilai, terminator baris apa pun, dan spasi putih	10240 Byte	Tidak
Total ukuran gabungan dari semua nilai header untuk API pribadi	8000 Byte	Tidak
Ukuran muatan	10 MB	Tidak
Tag per tahap	50	Tidak
Jumlah iterasi dalam satu <code>#foreach ... #end loop</code> dalam template pemetaan	1000	Tidak
Ukuran template pemetaan maksimum	300 KB	Tidak
Akses ukuran templat log	3 KB	Tidak
Metode panjang ARN	1600 byte. Jika metode Anda berisi parameter jalur dan klien menggunakan nilai yang melebihi panjang ARN, API Anda akan menampilkan respons. 414 Request URI too long	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Pengaturan pelambatan tingkat metode untuk tahap dalam rencana penggunaan	20	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center
Ukuran model per API	400 KB	Tidak
Jumlah sertifikat di truststore	1.000 sertifikat hingga 1 MB total ukuran objek.	Tidak
Batas waktu koneksi idle	310 detik	Tidak
Ukuran file definisi API maksimum saat menggunakan restapi:import atau restapi:put	6 MB	Tidak

* Anda tidak dapat mengatur batas waktu integrasi menjadi kurang dari 50 milidetik. Anda dapat meningkatkan batas waktu integrasi menjadi lebih dari 29 detik, tetapi ini mungkin memerlukan pengurangan kuota throttle tingkat Region untuk akun Anda.

Praktik terbaik untuk mengurangi kuota Anda

Praktik terbaik berikut dapat membantu mengurangi jumlah sumber daya Anda saat ini untuk menghindari peningkatan kuota Anda. Pastikan saran ini berfungsi untuk arsitektur API Anda.

APIs per Wilayah

Untuk mengurangi jumlah APIs per Wilayah, ekspor yang tidak terpakai APIs lalu hapus dari API Gateway. Untuk informasi selengkapnya, lihat [the section called “Ekspor REST API”](#).

Stage per API

Untuk mengurangi jumlah tahapan per API, pisahkan API Anda menjadi beberapa APIs.

Sumber daya per API

Gunakan {proxy+} jalur untuk mengurangi jumlah sumber daya. Untuk informasi selengkapnya, lihat [the section called “Siapkan integrasi proxy dengan sumber daya proxy”](#).

Pemetaan API

Untuk mengurangi jumlah pemetaan API untuk nama domain kustom, gunakan pemetaan API dengan satu level, seperti. /prod Pemetaan API dengan satu level tidak diperhitungkan dalam kuota pemetaan API.

Pengotorisasi untuk API

Untuk mengurangi jumlah otorisasi per API, gunakan kembali otorisasi di seluruh metode API.

Bagian dokumentasi per API

Untuk mengurangi jumlah bagian dokumentasi per API, gunakan ignore=documentation saat Anda mengimpor API. Untuk informasi selengkapnya, lihat [the section called “Impor dokumentasi API”](#).

Anda juga dapat menggunakan pewarisan konten untuk mengizinkan beberapa bagian dokumentasi mewarisi konten dari entitas API dengan spesifikasi yang lebih umum. Untuk informasi selengkapnya, lihat [the section called “Representasi dokumentasi API di API Gateway”](#).

Panjang, dalam karakter, kebijakan sumber daya API Gateway

Untuk mengurangi panjang kebijakan sumber daya, gunakan AWS WAFV2 untuk melindungi API Anda. Untuk informasi selengkapnya, lihat [the section called “AWS WAF”](#).

Jika kebijakan Anda berisi alamat IP, Anda juga dapat menggunakan rentang alih-alih nilai tertentu untuk menentukan alamat IP.

Paket penggunaan per kunci API

Untuk mengurangi jumlah paket penggunaan per kunci API, gunakan satu kunci API per paket penggunaan, dan kaitkan paket penggunaan Anda dengan beberapa APIs. Kami tidak menyarankan berbagi satu kunci API di beberapa paket penggunaan.

Kuota untuk mengonfigurasi dan menjalankan API HTTP di API Gateway

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan API HTTP di Amazon API Gateway. Jika kuota adalah per-API, itu hanya dapat ditingkatkan secara spesifik APIs, tidak untuk semua yang ada APIs di akun. Untuk informasi tentang kuota tingkat akun, lihat [Kuota dan catatan penting](#)

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Rute per API	300	Ya
Integrasi per API	300	Tidak
Batas waktu integrasi maksimum	30 detik	Tidak
Stage per API	10	Ya
Pemetaan API multi-level per domain	200 pemetaan API. Batas ini tidak termasuk pemetaan API dengan level tunggal, seperti/prod.	Tidak
Tag per tahap	50	Tidak
Total ukuran gabungan dari baris permintaan dan nilai header	10240 byte	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Ukuran muatan	10 MB	Tidak
Domain kustom per akun per Wilayah	120	Ya
Akses ukuran templat log	3 KB	Tidak
Entri CloudWatch log Amazon Logs	1 MB	Tidak
Pengotorisasi untuk API	10	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center
Pemirsa per otorisasi	50	Tidak
Lingkup per rute	10	Tidak
Batas waktu untuk titik akhir Set Kunci Web JSON	1500 ms	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Ukuran respons dari titik akhir JSON Web Key Set	150000 byte	Tidak
Batas waktu untuk titik akhir penemuan OpenID Connect	1500 ms	Tidak
Batas waktu untuk respons otorisasi Lambda	10000 ms	Tidak
Tautan VPC per akun per Wilayah	10	Ya
Subnet untuk tautan VPC	10	Ya
Variabel stage per stage	100	Tidak
Panjang, dalam karakter, dari kunci dalam variabel tahap	64	Tidak
Panjang, dalam karakter, dari nilai dalam variabel tahap	512	Tidak

Praktik terbaik untuk mengurangi kuota Anda

Praktik terbaik berikut dapat membantu mengurangi jumlah sumber daya Anda saat ini untuk menghindari peningkatan kuota Anda. Pastikan saran ini berfungsi untuk arsitektur API Anda.

APIs per Wilayah

Untuk mengurangi jumlah APIs per Wilayah, ekspor yang tidak terpakai APIs lalu hapus dari API Gateway. Untuk informasi selengkapnya, lihat [the section called “Ekspor”](#).

Stage per API

Untuk mengurangi jumlah tahapan per API, pisahkan API Anda menjadi beberapa APIs.

Pengotorisasi untuk API

Untuk mengurangi jumlah otorisasi per API, gunakan kembali otorisasi di seluruh metode API.

Kuota untuk mengonfigurasi portal di API Gateway

Kuota berikut berlaku untuk membuat portal di API Gateway. Untuk informasi selengkapnya, lihat [Portal API Gateway](#).

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Portal per akun	15	Ya
Produk portal per portal	200	Tidak
Produk portal per akun	500	Tidak
Halaman titik akhir REST produk per produk portal	40	Ya
Halaman produk per produk portal	40	Ya

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Ukuran logo	200 KB	Tidak
Ukuran halaman dokumentasi per produk REST halaman endpoint	32.000 karakter	Tidak
Ukuran halaman kustom untuk halaman produk	32.000 karakter	Tidak
Nama domain khusus per portal	1	Tidak
Pengotorisasi per portal	1	Tidak

Kuota untuk mengonfigurasi dan menjalankan API WebSocket Gateway

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan WebSocket API di Amazon API Gateway. Jika kuota adalah per-API, itu hanya dapat ditingkatkan secara spesifik APIs, tidak untuk semua yang ada APIs di akun. Untuk informasi tentang kuota tingkat akun, lihat [Kuota dan catatan penting](#)

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Koneksi baru per detik per akun (di	500	Ya

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
semua WebSocket APIs) per Wilayah		
Koneksi bersamaan	Tidak berlaku*	Tidak berlaku
AWS Lambda otorisasi per API	10	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center
AWS Lambda ukuran hasil otorisasi	8 KB	Tidak
Rute per API	300	Ya
Integrasi per API	300	Ya Untuk meningkatkan kuota ini, hubungi AWS Support Center

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Batas waktu integrasi	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	Tidak
Stage per API	10	Ya
WebSocket ukuran bingkai	32 KB	Tidak
Ukuran payload pesan	128 KB **	Tidak
Ukuran template pemetaan maksimum	300 KB	Tidak
Durasi koneksi untuk WebSocket API	2 jam	Tidak
Batas Waktu Koneksi Idle	10 menit	Tidak
Panjang, dalam karakter, URL untuk WebSocket API	4096	Tidak
Akses ukuran templat log	3 KB	Tidak

* API Gateway tidak memberlakukan kuota pada koneksi bersamaan. Jumlah maksimum koneksi bersamaan ditentukan oleh tingkat koneksi baru per detik dan durasi koneksi maksimum dua jam. Misalnya, dengan kuota default 500 koneksi baru per detik, jika klien terhubung dengan kecepatan maksimum selama dua jam, API Gateway dapat melayani hingga 3.600.000 koneksi bersamaan.

** Karena kuota WebSocket ukuran bingkai 32 KB, pesan yang lebih besar dari 32 KB harus dibagi menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Ini berlaku untuk `@connections` perintah. Jika pesan yang lebih besar (atau ukuran bingkai yang lebih besar) diterima, koneksi ditutup dengan kode 1009.

Catatan penting Amazon API Gateway

Bagian berikut merinci catatan yang mungkin memengaruhi penggunaan API Gateway Anda.

Topik

- [Catatan penting Amazon API Gateway untuk HTTP APIs](#)
- [Catatan penting Amazon API Gateway untuk HTTP dan WebSocket APIs](#)
- [Catatan penting Amazon API Gateway untuk REST dan WebSocket APIs](#)
- [Catatan penting Amazon API Gateway untuk WebSocket APIs](#)
- [Catatan penting Amazon API Gateway untuk REST APIs](#)

Catatan penting Amazon API Gateway untuk HTTP APIs

- HTTP APIs menerjemahkan X-Forwarded-* header masuk ke Forwarded header standar dan akan menambahkan IP keluar, Host, dan protokol.
- API Gateway menambahkan header Content-type ke permintaan Anda jika tidak ada payload dan Content-Length adalah 0.

Catatan penting Amazon API Gateway untuk HTTP dan WebSocket APIs

- Signature Version 4A tidak secara resmi didukung oleh Amazon API Gateway untuk HTTP dan WebSocket APIs.

Catatan penting Amazon API Gateway untuk REST dan WebSocket APIs

- API Gateway tidak mendukung berbagi nama domain khusus di seluruh REST dan WebSocket APIs.
- Nama panggung hanya dapat berisi karakter alfanumerik, tanda hubung, dan garis bawah. Panjang maksimum adalah 128 karakter.

- `/spingJalur` `/ping` dan disediakan untuk pemeriksaan kesehatan layanan. Penggunaan ini untuk sumber daya tingkat root API dengan domain khusus akan gagal menghasilkan hasil yang diharapkan.
- API Gateway saat ini membatasi peristiwa log hingga 1024 byte. Peristiwa log yang lebih besar dari 1024 byte, seperti badan permintaan dan respons, akan dipotong oleh API Gateway sebelum dikirim ke Log. CloudWatch
- CloudWatch Metrik saat ini membatasi nama dan nilai dimensi hingga 255 karakter XHTML yang valid. (Untuk informasi selengkapnya, lihat [Panduan CloudWatch Pengguna](#).) Nilai dimensi adalah fungsi dari nama yang ditentukan pengguna, termasuk nama API, nama label (tahap), dan nama sumber daya. Saat memilih nama-nama ini, berhati-hatilah untuk tidak melebihi batas CloudWatch Metrik.
- Ukuran maksimum template pemetaan adalah 300 KB.

Catatan penting Amazon API Gateway untuk WebSocket APIs

- API Gateway mendukung muatan pesan hingga 128 KB dengan ukuran bingkai maksimum 32 KB. Jika pesan melebihi 32 KB, Anda harus membaginya menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Jika pesan yang lebih besar diterima, koneksi ditutup dengan kode 1009.

Catatan penting Amazon API Gateway untuk REST APIs

- Karakter pipa teks biasa (|) dan karakter kurung kurawal ({atau}) tidak didukung untuk string kueri URL permintaan apa pun dan harus dikodekan URL.
- Karakter titik koma (;) tidak didukung untuk string kueri URL permintaan apa pun dan menghasilkan data yang dibagi.
- REST APIs mendekode parameter permintaan yang disandikan URL sebelum meneruskannya ke integrasi backend. Untuk parameter permintaan UTF-8, REST APIs memecahkan kode parameter dan kemudian meneruskannya sebagai unicode ke integrasi backend. REST APIs URL-encode persen karakter (%) sebelum meneruskannya ke integrasi backend.
- Saat menggunakan konsol API Gateway untuk menguji API, Anda mungkin mendapatkan respons “kesalahan titik akhir tidak dikenal” jika sertifikat yang ditandatangani sendiri ditampilkan ke backend, sertifikat perantara hilang dari rantai sertifikat, atau pengecualian terkait sertifikat lain yang tidak dapat dikenali yang dilemparkan oleh backend.

- Untuk API [Resource](#) atau [Method](#) entitas dengan integrasi pribadi, Anda harus menghapusnya setelah menghapus referensi kode keras dari file. [VpcLink](#) Jika tidak, Anda memiliki integrasi yang menggantung dan menerima kesalahan yang menyatakan bahwa tautan VPC masih digunakan bahkan ketika entitas Resource atau Method dihapus. Perilaku ini tidak berlaku ketika integrasi pribadi mereferensikan VpcLink melalui variabel tahap.
- Backend berikut mungkin tidak mendukung otentikasi klien SSL dengan cara yang kompatibel dengan API Gateway:
 - [NGINX](#)
 - [Heroku](#)
- API Gateway mendukung sebagian besar spesifikasi [OpenAPI 2.0 dan spesifikasi OpenAPI 3.0, dengan pengecualian](#) berikut:
 - Segmen jalur hanya dapat berisi karakter alfanumerik, garis bawah, tanda hubung, titik, koma, titik dua, dan kurawal kurawal. Parameter jalur harus segmen jalur terpisah. Misalnya, “resource/{path_parameter_name}” valid; “resource {path_parameter_name}” tidak.
 - Nama model hanya dapat berisi karakter alfanumerik.
 - Untuk parameter input, hanya atribut berikut yang didukung: name, in, required, type, description. Atribut lainnya diabaikan.
 - securitySchemesJenisnya, jika digunakan, harus apiKey. [Namun, otentikasi OAuth 2 dan HTTP Basic didukung melalui otorisasi Lambda; konfigurasi OpenAPI dicapai melalui ekstensi vendor.](#)
 - deprecatedBidang tidak didukung dan dijatuhkan di ekspor. APIs
 - Model API Gateway didefinisikan menggunakan [skema JSON draft 4, bukan skema](#) JSON yang digunakan oleh OpenAPI.
 - discriminatorParameter tidak didukung dalam objek skema apa pun.
 - exampleTag tidak didukung.
 - nullableBidang tidak didukung.
 - exclusiveMinimum tidak didukung oleh API Gateway.
 - minItemsTag maxItems dan tidak termasuk dalam [validasi permintaan sederhana](#). Untuk mengatasinya, perbarui model setelah impor sebelum melakukan validasi.
 - Untuk OpenAPI 3.0, Anda tidak dapat memiliki oneOf, anyOf, atau allOf yang digunakan \$ref untuk definisi dalam skema yang sama. Anda dapat langsung memasukkan skema atau menentukan sumber daya model API Gateway yang terpisah. Untuk informasi selengkapnya, lihat [the section called “Membuat model yang lebih kompleks”](#).

- oneOf tidak didukung untuk OpenAPI 2.0 atau pembuatan SDK.
- readOnly tidak didukung.
- \$ref tidak dapat digunakan untuk referensi file lain.
- Definisi respons "500": {"\$ref": "#/responses/UnexpectedError"} formulir tidak didukung di root dokumen OpenAPI. Untuk mengatasinya, ganti referensi dengan skema inline.
- Nomor Int64 tipe Int32 atau tidak didukung. Contoh ditunjukkan sebagai berikut:

```
"elementId": {  
  "description": "Working Element Id",  
  "format": "int32",  
  "type": "number"  
}
```

- Jenis format angka desimal ("format": "decimal") tidak didukung dalam definisi skema.
- Dalam respons metode, definisi skema harus dari tipe objek dan tidak dapat tipe primitif. Misalnya, "schema": { "type": "string"} tidak didukung. Namun, Anda dapat mengatasinya menggunakan jenis objek berikut:

```
"schema": {  
  "$ref": "#/definitions/StringResponse"  
}  
  
"definitions": {  
  "StringResponse": {  
    "type": "string"  
  }  
}
```

- API Gateway tidak menggunakan keamanan tingkat root yang ditentukan dalam spesifikasi OpenAPI. Oleh karena itu keamanan perlu didefinisikan pada tingkat operasi agar diterapkan dengan tepat.
- Kata default kunci tidak didukung.
- API Gateway memberlakukan batasan dan batasan berikut saat menangani metode dengan integrasi Lambda atau integrasi HTTP.
 - Nama header dan parameter kueri diproses dengan cara yang peka huruf besar/kecil.
 - Tabel berikut mencantumkan header yang mungkin dihapus, dipetakan ulang, atau diubah saat dikirim ke titik akhir integrasi atau dikirim kembali oleh titik akhir integrasi Anda. Dalam tabel ini:

- Remapped berarti bahwa nama header diubah dari *<string>* menjadi X-Amzn-Remapped-*<string>*.

Remapped Overwritten berarti bahwa nama header diubah dari *<string>* ke X-Amzn-Remapped-*<string>* dan nilainya ditimpa.

Nama header	Permintaan (http/http_proxy /lambda)	Tanggapan (http/http_proxy /lambda)
Age	Passthrough	Passthrough
Accept	Passthrough	Dropped/ Passthrough/ Passthrough
Accept-Charset	Passthrough	Passthrough
Accept-Encoding	Passthrough	Passthrough
Authorization	Passthrough *	Dipetakan ulang
Connection	Passthrough/Passthrough/Dropped	Dipetakan ulang
Content-Encoding	Passthrough/Dropped/Passthrough	Passthrough
Content-Length	Passthrough (dihasilkan berdasarkan tubuh)	Passthrough

Nama header	Permintaan (http/http_proxy /lambda)	Tanggapan (http/http_proxy /lambda)
Content-MD5	Jatuh	Dipetakan ulang
Content-Type	Passthrough	Passthrough
Date	Passthrough	Dipetakan Ulang Ditimpa
Expect	Jatuh	Jatuh
Host	Ditimpa ke titik akhir integrasi	Jatuh
Max-Forwards	Jatuh	Dipetakan ulang
Pragma	Passthrough	Passthrough
Proxy-Authorization	Jatuh	Jatuh
Range	Passthrough	Passthrough
Referer	Passthrough	Passthrough
Server	Jatuh	Dipetakan Ulang Ditimpa
TE	Jatuh	Jatuh

Nama header	Permintaan (http/http_proxy /lambda)	Tanggapan (http/http_proxy /lambda)
Transfer-Encoding	Dropped/Dropped/Exception	Jatuh
Trailer	Jatuh	Jatuh
Upgrade	Jatuh	Jatuh
User-Agent	Passthrough	Dipetakan ulang
Via	Dropped/Dropped/Passthrough	Passthrough/ Dropped/ Dropped
Warn	Passthrough	Passthrough
WWW-Authenticate	Jatuh	Dipetakan ulang

* Authorization Header dijatuhkan jika berisi tanda [tangan Versi Tanda Tangan 4](#) atau jika AWS_IAM otorisasi digunakan.

- Android SDK API yang dihasilkan oleh API Gateway menggunakan `java.net.HttpURLConnection` class tersebut. Kelas ini akan menampilkan pengecualian yang tidak tertangani, pada perangkat yang menjalankan Android 4.4 dan yang lebih lama, untuk respons 401 yang dihasilkan dari pemetaan ulang header ke. `WWW-Authenticate X-Amzn-Remapped-WWW-Authenticate`
- Tidak seperti Java, Android, dan iOS SDKs API yang dibuat oleh API Gateway, JavaScript SDK API yang dihasilkan oleh API Gateway tidak mendukung percobaan ulang untuk kesalahan tingkat 500.

- Pemanggilan pengujian metode menggunakan jenis konten default `application/json` dan mengabaikan spesifikasi jenis konten lainnya.
- Saat mengirim permintaan ke API dengan meneruskan `X-HTTP-Method-Override` header, API Gateway mengganti metode tersebut. Jadi untuk meneruskan header ke backend, header perlu ditambahkan ke permintaan integrasi.
- Jika permintaan berisi beberapa tipe media di `Accept` headernya, API Gateway hanya menghormati jenis `Accept` media pertama. Dalam situasi di mana Anda tidak dapat mengontrol urutan jenis `Accept` media dan jenis media konten biner Anda bukan yang pertama dalam daftar, Anda dapat menambahkan jenis `Accept` media pertama dalam `binaryMediaTypes` daftar API Anda, API Gateway akan mengembalikan konten Anda sebagai biner. Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir akan menerima file JPEG sebagai biner.
- Menyesuaikan respons gateway default untuk saat ini 413 `REQUEST_TOO_LARGE` tidak didukung.
- API Gateway menyertakan `Content-Type` header untuk semua respons integrasi. Secara default, jenis konten adalah `application/json`.

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir Amazon API Gateway. Untuk pemberitahuan tentang pembaruan dokumentasi ini, Anda dapat berlangganan umpan RSS dengan memilih tombol RSS di panel menu atas.

- Pembaruan dokumentasi terbaru: 02 Desember 2025

Perubahan	Deskripsi	Tanggal
Tambahkan API REST API Gateway sebagai target untuk Amazon Bedrock AgentCore Gateway	API Gateway sekarang mendukung penambahan REST API sebagai target untuk Amazon Bedrock AgentCore Gateway. Untuk informasi selengkapnya, lihat Menambahkan API Gateway REST API sebagai target Amazon Bedrock AgentCore Gateway .	Desember 2, 2025
Integrasi pribadi dengan Application Load Balancer	API Gateway sekarang mendukung integrasi pribadi dengan Application Load Balancers untuk REST. APIs Untuk informasi selengkapnya, lihat Integrasi pribadi untuk REST APIs .	November 21, 2025
Kebijakan keamanan untuk REST APIs dan nama domain khusus	API Gateway sekarang mendukung kebijakan keamanan TLS yang disempurnakan pada REST APIs dan nama domain khusus. Untuk informasi selengkapnya, lihat Kebijakan	November 19, 2025

[keamanan untuk REST APIs
di API Gateway.](#)

[Streaming respons untuk
REST APIs](#)

API Gateway REST APIs sekarang mendukung streaming muatan respons secara progresif ke klien saat tersedia. Untuk informasi selengkapnya, lihat [Streaming respons integrasi untuk integrasi proxy Anda](#).

November 19, 2025

[Portal pengembang](#)

API Gateway sekarang mendukung portal, yang merupakan lokasi terpusat bagi pelanggan Anda untuk menemukan dan menguji produk Anda APIs dan portal yang mewakili layanan atau fungsionalitas yang ingin Anda bagikan. Untuk informasi selengkapnya, lihat [Portal API Gateway](#).

November 19, 2025

[SIGv4a untuk REST APIs](#)

API Gateway sekarang mendukung penggunaan Signature Version 4a untuk REST APIs. Untuk informasi selengkapnya, lihat [model izin API Gateway untuk menjalankan API](#).

Agustus 19, 2025

[Aturan perutean untuk REST APIs](#)

API Gateway sekarang mendukung aturan perutean untuk REST APIs menggunakan nama domain khusus. Anda dapat merutekan permintaan masuk secara dinamis berdasarkan nilai header HTTP, jalur dasar URL, atau kombinasi keduanya. Untuk informasi selengkapnya, lihat [Mengirim lalu lintas ke nama domain kustom Anda](#). APIs

Juni 3, 2025

[Menambahkan dukungan untuk titik akhir dual-stack \(IPv4 dan IPv6\)](#)

API Gateway sekarang mendukung titik akhir dual-stack untuk REST, HTTP, dan WebSocket APIs, dan nama domain kustom. Untuk informasi selengkapnya, lihat [Jenis alamat IP](#).

Maret 28, 2025

[Nama domain khusus untuk pribadi APIs](#)

Ditambahkan dukungan untuk nama domain kustom untuk pribadi APIs. Untuk informasi selengkapnya, lihat [Nama domain khusus untuk pribadi APIs](#).

November 21, 2024

[Menambahkan dukungan untuk TLS 1.3](#)

API Gateway sekarang mendukung TLS 1.3 di REST Regional APIs, HTTP APIs, dan WebSocket APIs. Untuk informasi selengkapnya, lihat [Memilih kebijakan keamanan untuk domain kustom Anda di API Gateway](#).

Februari 15, 2024

[Pembaruan API REST dan konsol WebSocket API](#)

Informasi konsol yang diperbarui untuk REST APIs dan WebSocket APIs.

Desember 10, 2023

[Pembaruan dokumentasi](#)

Memperbarui informasi konseptual dan membuat tutorial baru untuk transformasi data dan meminta topik validasi untuk API Gateway REST. APIs Untuk informasi selengkapnya, lihat [Menggunakan validasi permintaan di API Gateway](#) dan [Menyiapkan transformasi data untuk REST API](#).

22 Juni 2023

[Konfigurasi failover DNS untuk API Gateway Multi-wilayah](#)

Menambahkan dukungan untuk menggunakan pemeriksaan kesehatan Amazon Route 53 untuk mengontrol failover DNS dari API REST API Gateway API di primer Wilayah AWS ke satu di Wilayah sekunder. Untuk informasi selengkapnya, lihat [Mengonfigurasi pemeriksaan kesehatan khusus untuk failover DNS](#).

31 Oktober 2022

[Pembaruan dokumentasi](#)

Ringkasan fitur inti yang diperbarui untuk REST API dan HTTP API. APIs Untuk informasi selengkapnya, lihat [Memilih antara REST API dan HTTP API APIs](#).

31 Mei 2022

Pembaruan kebijakan terkelola	Menambahkan <code>acm:GetCertificate</code> dukungan pada <code>AWSServiceRoleForAPIGateway</code> kebijakan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk API Gateway .	12 Juli 2021
Pemetaan parameter untuk HTTP APIs	Ditambahkan dukungan untuk pemetaan parameter untuk HTTP APIs. Untuk informasi selengkapnya, lihat Mengubah permintaan dan respons API .	7 Januari 2021
Nonaktifkan titik akhir default untuk REST API	Menambahkan dukungan untuk menonaktifkan titik akhir default untuk REST APIs. Untuk informasi selengkapnya, lihat Menonaktifkan titik akhir default untuk REST API .	29 Oktober 2020
Otentikasi TLS timbal balik	Menambahkan dukungan untuk otentikasi TLS timbal balik untuk REST APIs dan HTTP APIs. Untuk informasi selengkapnya, lihat Mengonfigurasi autentikasi TLS timbal balik untuk REST API dan Mengonfigurasi otentikasi TLS timbal balik untuk API HTTP .	17 September 2020

[AWS Lambda Otorisasi HTTP API](#)

Ditambahkan dukungan untuk AWS Lambda otorisasi untuk HTTP APIs. Untuk informasi selengkapnya, lihat [Bekerja dengan AWS Lambda otorisasi untuk HTTP APIs](#).

9 September 2020

[Integrasi AWS layanan HTTP API](#)

Menambahkan dukungan untuk integrasi AWS layanan untuk HTTP APIs. Untuk informasi selengkapnya, lihat [Bekerja dengan integrasi AWS layanan untuk HTTP APIs](#).

20 Agustus 2020

[Domain kustom wildcard API HTTP](#)

Ditambahkan dukungan untuk nama domain kustom wildcard untuk HTTP APIs. Untuk informasi selengkapnya, lihat [Nama domain kustom Wildcard](#).

10 Agustus 2020

[Perbaikan portal pengembang tanpa server](#)

Menambahkan manajemen pengguna ke panel administrator dan dukungan untuk mengekspor definisi API. Untuk informasi selengkapnya, lihat [Menggunakan portal developer tanpa server untuk membuat katalog API Gateway Anda](#). APIs

25 Juni 2020

WebSocket Sec-WebSocket-Protocol Dukungan API	Menambahkan dukungan untuk Sec-WebSocket-Protocol lapangan. Untuk informasi selengkapnya, lihat Menyiapkan rute \$connect yang memerlukan WebSocket subprotokol .	16 Juni 2020
Ekspor API HTTP	Menambahkan dukungan untuk mengekspor definisi OpenAPI 3.0 HTTP. APIs Untuk informasi selengkapnya, lihat Mengekspor API HTTP dari API Gateway .	20 April 2020
Dokumentasi keamanan	Menambahkan dokumentasi keamanan. Untuk informasi selengkapnya, lihat Keamanan di Amazon API Gateway .	31 Maret 2020
Dokumentasi yang direorganisasi	Menata ulang panduan pengembang.	12 Maret 2020
Ketersediaan umum HTTP API	Dirilis HTTP APIs dalam ketersediaan umum. Untuk informasi selengkapnya, lihat Bekerja dengan HTTP APIs .	12 Maret 2020
Pencatatan API HTTP	Menambahkan dukungan untuk \$context.integrationErrorMessage di log API HTTP. Untuk informasi selengkapnya, lihat Variabel Pencatatan API HTTP .	26 Februari 2020

AWS variabel untuk impor OpenAPI	Menambahkan dukungan untuk AWS variabel dalam definisi OpenAPI. Untuk informasi selengkapnya, lihat AWS Variabel untuk Impor OpenAPI .	17 Februari 2020
HTTP APIs	Dirilis HTTP APIs dalam versi beta. Untuk informasi selengkapnya, lihat HTTP APIs .	4 Desember 2019
Nama domain kustom wildcard	Ditambahkan dukungan untuk nama domain kustom wildcard. Untuk informasi selengkapnya, lihat Nama Domain Kustom Wildcard .	Oktober 21, 2019
Pencatatan Amazon Data Firehose	Menambahkan dukungan untuk Amazon Data Firehose sebagai tujuan untuk mengakses data logging. Untuk informasi selengkapnya, lihat Menggunakan Amazon Data Firehose sebagai Tujuan untuk Pencatatan Akses Gateway API .	15 Oktober 2019
Alias Route53 untuk memanggil pribadi APIs	Menambahkan dukungan untuk catatan DNS alias Route53 tambahan untuk memanggil pribadi. APIs Untuk informasi selengkapnya, lihat Mengakses API Pribadi Anda Menggunakan Alias Route53 .	18 September 2019

Kontrol akses berbasis tag untuk WebSocket APIs	Menambahkan dukungan untuk kontrol akses berbasis tag untuk WebSocket APIs. Untuk informasi selengkapnya, lihat Sumber Daya API Gateway yang Dapat Ditandai .	27 Juni 2019
Pilihan versi TLS untuk domain kustom	Menambahkan dukungan untuk pemilihan versi Transport Layer Security (TLS) untuk APIs yang diterapkan ke domain khusus. Lihat catatan di Pilih Versi TLS Minimum untuk Domain Kustom di API Gateway .	20 Juni 2019
Kebijakan titik akhir VPC untuk pribadi APIs	Menambahkan dukungan untuk meningkatkan keamanan pribadi APIs dengan melampirkan kebijakan titik akhir ke titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Menggunakan Kebijakan Titik Akhir VPC untuk Pribadi di API APIs Gateway .	4 Juni 2019
Dokumentasi diperbarui	Menulis Ulang Memulai dengan Amazon API Gateway . Pindah tutorial ke Amazon API Gateway Tutorial .	29 Mei 2019

Kontrol akses berbasis tag untuk REST APIs	Menambahkan dukungan untuk kontrol akses berbasis tag untuk REST APIs. Untuk informasi selengkapnya, lihat Menggunakan Tag dengan Kebijakan IAM untuk Mengontrol Akses ke Sumber Daya API Gateway .	23 Mei 2019
Dokumentasi diperbarui	Menulis ulang topik 6: Apa itu Amazon API Gateway? , Tutorial: Membangun API dengan Integrasi Proxy HTTP , Tutorial: Membuat Calc REST API dengan Tiga Integrasi Non-Proxy , Template Pemetaan API Gateway dan Access Logging Variabel Referensi , Gunakan API Gateway Lambda Authorizers , dan Aktifkan CORS untuk API Gateway REST API Resource.	5 April 2019
Perbaikan portal pengembang tanpa server	Menambahkan panel administrator dan peningkatan lainnya untuk memudahkan publikasi APIs di portal pengembangan Amazon API Gateway. Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog Anda APIs .	28 Maret 2019

Support untuk AWS Config	Menambahkan dukungan untuk AWS Config. Untuk informasi selengkapnya, lihat Memantau Konfigurasi API Gateway API dengan AWS Config .	20 Maret 2019
Support untuk CloudFormation	Menambahkan API Gateway V2 API ke referensi CloudFormation template. Untuk informasi selengkapnya, lihat Referensi Jenis Sumber Daya Amazon API Gateway V2 .	7 Februari 2019
Support untuk WebSocket APIs	Menambahkan dukungan untuk WebSocket APIs. Untuk informasi selengkapnya, lihat Membuat WebSocket API di Amazon API Gateway .	18 Desember 2018
Portal pengembang tanpa server tersedia melalui AWS Serverless Application Repository	Aplikasi tanpa server portal pengembang Amazon API Gateway sekarang tersedia dari AWS Serverless Application Repository (selain GitHub). Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog API Gateway Anda APIs .	16 November 2018
Support untuk AWS WAF	Ditambahkan dukungan untuk AWS WAF (Web Application Firewall). Untuk informasi selengkapnya, lihat Mengontrol Akses ke Menggunakan API AWS WAF .	5 November 2018

Portal pengembang tanpa server	Amazon API Gateway sekarang menyediakan portal pengembang yang sepenuhnya dapat disesuaikan sebagai aplikasi tanpa server yang dapat Anda terapkan untuk menerbitkan API Gateway Anda. APIs Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog API Gateway Anda APIs .	29 Oktober 2018
Support untuk header multi-nilai dan parameter string kueri	Amazon API Gateway sekarang mendukung beberapa header dan parameter string kueri yang memiliki nama yang sama. Untuk informasi selengkapnya, lihat Support for Multi-Value Header dan Query String Parameters .	4 Oktober 2018
Dukungan OpenAPI	Amazon API Gateway sekarang mendukung OpenAPI 3.0 serta OpenAPI (Swagger) 2.0.	27 September 2018
Dokumentasi diperbarui	Menambahkan topik baru: Bagaimana Kebijakan Sumber Daya Amazon API Gateway Mempengaruhi Alur Kerja Otorisasi .	27 September 2018

[AWS X-Ray Integrasi aktif](#)

Anda sekarang dapat menggunakan AWS X-Ray untuk melacak dan menganalisis latensi dalam permintaan pengguna saat mereka melakukan perjalanan melalui layanan APIs yang mendasarinya. Untuk informasi selengkapnya, lihat [Melacak Eksekusi API Gateway API dengan AWS X-Ray](#).

6 September 2018

[Perbaikan caching](#)

Hanya GET metode yang akan mengaktifkan caching secara default saat Anda mengaktifkan caching untuk tahap API. Ini membantu memastikan keamanan API Anda. Anda dapat mengaktifkan caching untuk metode lain dengan mengganti pengaturan metode. Untuk informasi selengkapnya, lihat [Mengaktifkan Caching API untuk Meningkatkan Responsivitas](#).

20 Agustus 2018

[Batas layanan direvisi](#)

Beberapa batasan telah direvisi: Peningkatan jumlah APIs per akun. Peningkatan batas tarif API untuk Create/Import/Deploy APIs. Mengoreksi beberapa tarif dari per menit ke per detik. Untuk informasi selengkapnya, lihat [Batasan-batasan](#).

13 Juli 2018

[Mengganti parameter dan header permintaan dan respons API](#)

Menambahkan dukungan untuk mengganti header permintaan, string kueri, dan jalur, serta header respons dan kode status. Untuk informasi selengkapnya, lihat [Menggunakan Template Pemetaan untuk Mengganti Parameter dan Header Permintaan dan Respons API](#).

12 Juli 2018

[Pelambatan tingkat metode untuk paket penggunaan](#)

Menambahkan dukungan untuk menyetel batas pelambatan per metode default, serta menetapkan batas pelambatan untuk metode API individual dalam pengaturan paket penggunaan. Pengaturan ini merupakan tambahan dari pembatasan tingkat akun yang ada dan batas pelambatan tingkat metode default yang dapat Anda atur dalam pengaturan tahap. Untuk informasi selengkapnya, lihat [Permintaan API Throttle untuk Throughput yang Lebih Baik](#).

11 Juli 2018

[Pemberitahuan pembaruan Panduan Pengembang API Gateway sekarang tersedia melalui RSS](#)

Versi HTML Panduan Pengembang API Gateway sekarang mendukung umpan RSS pembaruan yang didokumentasikan di halaman Riwayat Dokumen ini. Umpan RSS mencakup pembaruan yang dibuat 27 Juni 2018, dan yang lebih baru. Pembaruan yang diumumkan sebelumnya masih tersedia di halaman ini. Gunakan tombol RSS di panel menu atas untuk berlangganan feed.

27 Juni 2018

Pembaruan lebih awal

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengembang API Gateway sebelum 27 Juni 2018.

Ubah	Deskripsi	Tanggal diubah
Pribadi APIs	Menambahkan dukungan untuk pribadi APIs , yang Anda paparkan melalui titik akhir VPC antarmuka . Lalu lintas ke pribadi Anda APIs tidak meninggalkan jaringan Amazon; itu terisolasi dari internet publik.	14 Juni 2018
Otorisasi dan Integrasi Lambda Lintas Akun dan Otorisasi Kumpulan Pengguna Amazon Cognito Lintas Akun	Gunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai fungsi otorisasi Lambda atau sebagai backend integrasi API. Atau gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi. Akun lain dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Lihat informasi selengkapnya di the section called “Konfigurasi otorisasi Lambda lintas akun” , the section called “Tutorial: Buat REST API dengan integrasi proxy	2 April 2018

Ubah	Deskripsi	Tanggal diubah
	Lambda lintas akun ”, dan the section called “Konfigurasi otorisasi Amazon Cognito lintas akun untuk REST API” .	
Kebijakan Sumber Daya untuk APIs	Gunakan kebijakan sumber daya API Gateway untuk memungkinkan pengguna dari AWS akun lain mengakses API Anda dengan aman atau mengizinkan API dipanggil hanya dari rentang alamat IP sumber tertentu atau blok CIDR. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan sumber daya API Gateway” .	Selasa, 02 April 2018
Penandaan untuk sumber daya API Gateway	Tandai tahap API dengan hingga 50 tag untuk alokasi biaya permintaan API dan caching di API Gateway. Untuk mengetahui informasi selengkapnya, lihat the section called “Siapkan tag” .	19 Desember 2017
Kompresi muatan dan dekompresi	Aktifkan panggilan API Anda dengan muatan terkompresi menggunakan salah satu pengkodean konten yang didukung. Muatan terkompresi tunduk pada pemetaan jika templat pemetaan tubuh ditentukan. Untuk informasi selengkapnya, lihat the section called “Pengkodean konten” .	19 Desember 2017
Kunci API yang bersumber dari otorisasi khusus	Kembalikan kunci API dari otorisasi khusus ke API Gateway untuk menerapkan rencana penggunaan metode API yang memerlukan kunci tersebut. Untuk informasi selengkapnya, lihat the section called “Pilih sumber kunci API di API Gateway” .	19 Desember 2017
Otorisasi dengan OAuth 2 cakupan	Aktifkan otorisasi pemanggilan metode dengan menggunakan OAuth 2 cakupan dengan otorisasi. COGNITO_USER_POOL Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API” .	14 Desember 2017

Ubah	Deskripsi	Tanggal diubah
Integrasi Pribadi dan Tautan VPC	Buat API dengan integrasi pribadi API Gateway untuk memberi klien akses ke HTTP/HTTPS sumber daya di VPC Amazon dari luar VPC melalui sumber daya. VpcLink Untuk informasi selengkapnya, lihat the section called “Tutorial: Buat REST API dengan integrasi pribadi” dan the section called “Siapkan integrasi pribadi” .	30 November 2017
Menerapkan Canary untuk pengujian API	Tambahkan rilis canary ke penerapan API yang ada untuk menguji versi API yang lebih baru sambil menjaga versi saat ini beroperasi pada tahap yang sama. Anda dapat mengatur persentase lalu lintas tahap untuk rilis kenari dan mengaktifkan eksekusi khusus kenari dan akses yang dicatat di log Log terpisah. CloudWatch Untuk informasi selengkapnya, lihat the section called “Siapkan penerapan rilis kenari” .	Selasa, 28 Nopember 2017
Akses Pencatatan	Log akses klien ke API Anda dengan data yang berasal dari variabel \$context dalam format yang Anda pilih. Untuk informasi selengkapnya, lihat the section called “CloudWatch log” .	21 November 2017
Ruby SDK dari sebuah API	Buat Ruby SDK untuk API Anda dan gunakan untuk menjalankan metode API Anda. Untuk informasi selengkapnya, lihat the section called “Menghasilkan Ruby SDK API di API Gateway” dan the section called “Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API” .	20 November 2017

Ubah	Deskripsi	Tanggal diubah
Titik akhir API regional	Tentukan titik akhir API regional untuk membuat API bagi klien non-seluler. Klien non-seluler, seperti EC2 instance, berjalan di AWS Wilayah yang sama tempat API diterapkan. Seperti halnya API yang dioptimalkan tepi, Anda dapat membuat nama domain khusus untuk API regional. Untuk informasi selengkapnya, lihat the section called “Jenis titik akhir API Gateway” dan the section called “Siapkan nama domain kustom Regional” .	2 November 2017
Authorizer permintaan kustom	Gunakan otorisasi permintaan khusus untuk menyediakan informasi otentikasi pengguna dalam parameter permintaan untuk mengotorisasi panggilan metode API. Parameter permintaan mencakup header dan parameter string kueri serta variabel tahap dan konteks. Untuk informasi selengkapnya, lihat Gunakan otorisasi API Gateway Lambda .	15 September 2017
Menyesuaikan tanggapan gateway	Sesuaikan respons gateway yang dihasilkan API Gateway ke permintaan API yang gagal mencapai backend integrasi. Pesan gateway yang disesuaikan dapat memberi pemanggil pesan kesalahan khusus API, termasuk mengembalikan header CORS yang diperlukan, atau dapat mengubah data respons gateway ke format pertukaran eksternal. Untuk informasi selengkapnya, lihat Menyiapkan respons gateway untuk menyesuaikan respons kesalahan .	6 Juni 2017
Memetakan properti kesalahan kustom Lambda ke header respons metode	Petakan properti kesalahan kustom individual yang dikembalikan dari Lambda ke parameter header respons metode menggunakan <code>integration.response.body</code> parameter, mengandalkan API Gateway untuk deserialisasi objek kesalahan kustom yang dirangkai saat dijalankan. Untuk informasi selengkapnya, lihat Menangani kesalahan Lambda khusus di API Gateway .	6 Juni 2017

Ubah	Deskripsi	Tanggal diubah
Batas pelambatan meningkat	Tingkatkan batas tingkat permintaan kondisi tunak tingkat akun menjadi 10.000 permintaan per detik (rps) dan batas burst menjadi 5000 permintaan bersamaan. Untuk informasi selengkapnya, lihat Throttle request ke REST APIs Anda untuk throughput yang lebih baik di API Gateway .	6 Juni 2017
Memvalidasi permintaan metode	Konfigurasi validator permintaan dasar pada level API atau level metode sehingga API Gateway dapat memvalidasi permintaan yang masuk. API Gateway memverifikasi bahwa parameter yang diperlukan disetel dan tidak kosong, dan memverifikasi bahwa format muatan yang berlaku sesuai dengan model yang dikonfigurasi. Untuk informasi selengkapnya, lihat Minta validasi untuk REST APIs di API Gateway .	11 April 2017
Integrasi dengan ACM	Gunakan Sertifikat ACM untuk nama domain kustom API Anda. Anda dapat membuat sertifikat AWS Certificate Manager atau mengimpor sertifikat berformat PEM yang ada ke ACM. Anda kemudian merujuk ke ARN sertifikat saat menetapkan nama domain khusus untuk Anda. APIs Untuk informasi selengkapnya, lihat Nama domain khusus untuk REST publik APIs di API Gateway .	Maret 9, 2017
Menghasilkan dan memanggil Java SDK dari API	Biarkan API Gateway menghasilkan Java SDK untuk API Anda dan gunakan SDK untuk memanggil API di klien Java Anda. Untuk informasi selengkapnya, lihat Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API .	Januari 13, 2017

Ubah	Deskripsi	Tanggal diubah
Integrasi dengan AWS Marketplace	Jual API Anda dalam paket penggunaan sebagai produk SaaS. AWS Marketplace Gunakan AWS Marketplace untuk memperluas jangkauan API Anda. AWS Marketplace Andalkan tagihan pelanggan atas nama Anda. Biarkan API Gateway menangani otorisasi pengguna dan pengukuran penggunaan. Untuk informasi selengkapnya, lihat Jual APIs sebagai SaaS Anda .	1 Desember 2016
Mengaktifkan Documentation Support untuk API	Tambahkan dokumentasi untuk entitas API dalam DocumentationPart sumber daya di API Gateway. Kaitkan snapshot <code>DocumentationPart</code> instance koleksi dengan tahap API untuk membuat file. DocumentationVersion Publikasikan dokumentasi API dengan mengekspor versi dokumentasi ke file eksternal, seperti file Swagger. Untuk informasi selengkapnya, lihat Dokumentasi untuk REST APIs di API Gateway .	1 Desember 2016
Authorizer kustom yang diperbarui	Fungsi Lambda otorisasi pelanggan sekarang mengembalik pengenal utama pemanggil. Fungsi ini juga dapat mengembalikan informasi lain sebagai pasangan kunci-nilai context peta dan kebijakan IAM. Untuk informasi selengkapnya, lihat Keluaran dari otorisasi API Gateway Lambda .	1 Desember 2016
Mendukung muatan biner	Tetapkan binaryMediaTypes pada API Anda untuk mendukung muatan biner permintaan atau respons. Tetapkan <code>contentHandling</code> properti pada Integrasi atau IntegrationResponse untuk menentukan apakah akan menangani payload biner sebagai gumpalan biner asli, sebagai string Base64-enocded, atau sebagai passthrough tanpa modifikasi. Untuk informasi selengkapnya, lihat Jenis media biner untuk REST APIs di API Gateway .	17 November 2016

Ubah	Deskripsi	Tanggal diubah
Mengaktifkan integrasi proxy dengan backend HTTP atau Lambda melalui sumber daya proxy API.	Buat sumber daya proxy dengan parameter jalur serakah dari formulir {proxy+} dan metode catch-allANY. Sumber daya proxy terintegrasi dengan backend HTTP atau Lambda masing-masing menggunakan integrasi proxy HTTP atau Lambda. Untuk informasi selengkapnya, lihat Siapkan integrasi proxy dengan sumber daya proxy .	20 September 2016
Memperluas pilihan APIs di API Gateway sebagai penawaran produk untuk pelanggan Anda dengan menyediakan satu atau beberapa paket penggunaan.	Buat paket penggunaan di API Gateway untuk memungkinkan klien API terpilih mengakses tahapan API tertentu dengan tarif dan kuota permintaan yang disepakati. Untuk informasi selengkapnya, lihat Paket penggunaan dan kunci API untuk REST APIs di API Gateway .	11 Agustus, 2016
Mengaktifkan otorisasi tingkat metode dengan kumpulan pengguna di Amazon Cognito	Buat kumpulan pengguna di Amazon Cognito dan gunakan sebagai penyedia identitas Anda sendiri. Anda dapat mengonfigurasi kumpulan pengguna sebagai otorisasi tingkat metode untuk memberikan akses bagi pengguna yang terdaftar di kumpulan pengguna. Untuk informasi selengkapnya, lihat Kontrol akses ke REST APIs menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi .	28 Juli 2016
Mengaktifkan CloudWatch metrik dan dimensi Amazon di bawah namespace. AWS/ApiGateway	Metrik API Gateway sekarang distandarisasi di bawah namespace. CloudWatch AWS/ApiGateway Anda dapat melihatnya di konsol API Gateway dan CloudWatch konsol Amazon. Untuk informasi selengkapnya, lihat Dimensi dan metrik Amazon API Gateway .	28 Juli 2016

Ubah	Deskripsi	Tanggal diubah
Mengaktifkan rotasi sertifikat untuk nama domain kustom	Rotasi sertifikat memungkinkan Anda untuk mengunggah dan memperbarui sertifikat kedaluwarsa untuk nama domain kustom. Untuk informasi selengkapnya, lihat Putar sertifikat yang diimpor ke ACM .	27 April 2016
Mendokumentasikan perubahan untuk konsol Amazon API Gateway yang diperbarui.	Pelajari cara membuat dan menyiapkan API menggunakan konsol API Gateway yang diperbarui. Untuk informasi selengkapnya, lihat Tutorial: Buat REST API dengan mengimpor contoh dan Tutorial: Membuat REST API dengan integrasi non-proxy HTTP .	5 April 2016
Mengaktifkan fitur Import API untuk membuat API baru atau memperbarui API yang sudah ada dari definisi API eksternal.	Dengan fitur API Impor, Anda dapat membuat API baru atau memperbarui API yang sudah ada dengan mengunggah definisi API eksternal yang dinyatakan dalam Swagger 2.0 dengan ekstensi API Gateway. Untuk informasi selengkapnya tentang API Impor, lihat Kembangkan REST API menggunakan OpenAPI di API Gateway .	5 April 2016
Mengekspos <code>\$input.body</code> variabel untuk mengakses muatan mentah sebagai string dan <code>\$util.parseJson()</code> fungsi untuk mengubah string JSON menjadi objek JSON dalam template pemetaan.	Untuk informasi selengkapnya tentang <code>\$input.body</code> dan <code>\$util.parseJson()</code> , lihat Variabel untuk transformasi data untuk API Gateway .	5 April 2016

Ubah	Deskripsi	Tanggal diubah
Mengaktifkan permintaan klien dengan pembatalan cache tingkat metode, dan meningkatkan manajemen pembatasan permintaan.	Siram cache tingkat tahap API dan batalkan entri cache individual. Untuk informasi selengkapnya, lihat Siram cache tahap API di API Gateway dan Membatalkan entri cache API Gateway . Tingkatkan pengalaman konsol untuk mengelola pembatasan permintaan API. Untuk informasi selengkapnya, lihat Throttle request ke REST APIs Anda untuk throughput yang lebih baik di API Gateway .	25 Maret 2016
Mengaktifkan dan memanggil API Gateway API menggunakan otorisasi khusus	Buat dan konfigurasi AWS Lambda fungsi untuk mengimplementasikan otorisasi kustom. Fungsi ini mengembalikan dokumen kebijakan IAM yang memberikan izin Izinkan atau Tolak ke permintaan klien API Gateway API. Untuk informasi selengkapnya, lihat Gunakan otorisasi API Gateway Lambda .	Selasa, 11 Februari 2016
Mengimpor dan mengekspor API Gateway API menggunakan file definisi Swagger dan ekstensi	Buat dan perbarui API Gateway API Anda menggunakan spesifikasi Swagger dengan ekstensi API Gateway. Impor definisi Swagger menggunakan API Gateway Importer. Ekspor API Gateway API ke file definisi Swagger menggunakan konsol API Gateway atau API Gateway Export API. Untuk informasi selengkapnya, lihat Kembangkan REST API menggunakan OpenAPI di API Gateway dan Ekspor REST API dari API Gateway .	18 Desember 2015
Memetakan permintaan atau badan respons atau bidang JSON badan untuk meminta atau menanggapi parameter.	Metode peta meminta badan atau bidang JSON ke dalam jalur permintaan integrasi, string kueri, atau header. Petakan badan respons integrasi atau bidang JSON-nya ke dalam header respons permintaan. Untuk informasi selengkapnya, lihat Contoh pemetaan parameter untuk REST API di API Gateway .	18 Desember 2015

Ubah	Deskripsi	Tanggal diubah
Bekerja dengan Variabel Tahap di Amazon API Gateway	Pelajari cara mengaitkan atribut konfigurasi dengan tahap penerapan API di Amazon API Gateway. Untuk informasi selengkapnya, lihat Menggunakan variabel stage untuk REST API di API Gateway .	November 5, 2015
Cara: Aktifkan CORS untuk Metode	Sekarang lebih mudah untuk mengaktifkan berbagi sumber daya lintas asal (CORS) untuk metode di Amazon API Gateway. Untuk informasi selengkapnya, lihat CORS untuk REST APIs di API Gateway .	November 3, 2015
Cara: Menggunakan Otentikasi SSL Sisi Klien	Gunakan Amazon API Gateway untuk menghasilkan sertifikat SSL yang dapat Anda gunakan untuk mengautentikasi panggilan ke backend HTTP Anda. Untuk informasi selengkapnya, lihat Buat dan konfigurasi sertifikat SSL untuk otentikasi backend di API Gateway .	September 22, 2015
Integrasi metode tiruan	Pelajari cara mengintegrasikan API tiruan dengan Amazon API Gateway . Fitur ini memungkinkan pengembang untuk menghasilkan respons API dari API Gateway secara langsung tanpa perlu backend integrasi akhir sebelumnya.	1 September 2015
Dukungan Identitas Amazon Cognito	Amazon API Gateway telah memperluas cakupan <code>\$context</code> variabel sehingga sekarang mengembalikan informasi tentang Identitas Amazon Cognito ketika permintaan ditandatangani dengan kredensial Amazon Cognito. Selain itu, kami telah menambahkan <code>\$util</code> variabel untuk melarikan diri karakter dalam JavaScript dan pengkodean URLs dan string. Untuk informasi selengkapnya, lihat Variabel untuk transformasi data untuk API Gateway .	Agustus 28, 2015

Ubah	Deskripsi	Tanggal diubah
Integrasi kesombongan	Gunakan alat impor Swagger GitHub untuk mengimpor definisi Swagger API ke Amazon API Gateway. Pelajari lebih lanjut Ekstensi OpenAPI untuk API Gateway cara membuat APIs dan menerapkan serta metode menggunakan alat impor. Dengan alat importir Swagger Anda juga dapat memperbarui yang ada. APIs	21 Juli 2015
Referensi Template Pemetaan	Baca tentang \$input parameter dan fungsinya di Variabel untuk transformasi data untuk API Gateway .	Juli 18, 2015
Rilis publik awal	Ini adalah rilis publik awal dari Panduan Pengembang API Gateway.	9 Juli 2015

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.