Leitfaden zur Implementierung

Verteilte Lasttests auf AWS

Verteilte Lasttests auf AWS: Leitfaden zur Implementierung

Table of Contents

Automatisieren Sie das Testen Ihrer Softwareanwendungen in großem Maßstab

Datum der Veröffentlichung: Dezember 2025

Distributed Load Testing auf AWS hilft Ihnen dabei, Leistungstests Ihrer Softwareanwendungen im großen Maßstab zu automatisieren, um Engpässe zu identifizieren, bevor Sie Ihre Anwendung veröffentlichen. Diese Lösung simuliert Tausende von verbundenen Benutzern, die HTTP-Anfragen mit kontinuierlicher Geschwindigkeit generieren, ohne dass Server bereitgestellt werden müssen.

Diese Lösung nutzt Amazon Elastic Container Service (Amazon ECS) auf AWS Fargate, um Container bereitzustellen, auf denen Ihre Lasttestsimulationen ausgeführt werden, und bietet die folgenden Funktionen:

- Stellen Sie Amazon ECS auf AWS Fargate-Containern bereit, die unabhängig voneinander ausgeführt werden, um die Ladekapazität Ihrer Anwendung zu testen.
- Simulieren Sie Zehntausende von gleichzeitigen Benutzern in mehreren AWS-Regionen, die kontinuierlich Anfragen generieren.
- Passen Sie Ihre Anwendungstests mithilfe von K6 JMeter, Locust-Testskripten oder einer einfachen HTTP-Endpunktkonfiguration an.
- Planen Sie Lasttests so, dass sie sofort, zu einem future Zeitpunkt oder nach einem wiederkehrenden Zeitplan ausgeführt werden.
- Führen Sie mehrere Lasttests gleichzeitig in verschiedenen Szenarien und Regionen durch.

Dieser Implementierungsleitfaden bietet einen Überblick über die Lösung Distributed Load Testing on AWS, ihre Referenzarchitektur und Komponenten, Überlegungen zur Planung der Bereitstellung und Konfigurationsschritte für die Bereitstellung der Lösung in der Amazon Web Services (AWS) -Cloud. Es enthält Links zu einer CloudFormationAWS-Vorlage, mit der die AWS-Services gestartet und konfiguriert werden, die für die Bereitstellung dieser Lösung erforderlich sind, wobei die bewährten AWS-Methoden für Sicherheit und Verfügbarkeit verwendet werden.

Zu den Zielgruppen für die Nutzung der Funktionen und Fähigkeiten dieser Lösung in ihrer Umgebung gehören IT-Infrastrukturarchitekten, Administratoren und DevOps Fachleute, die über praktische Erfahrung in der Architektur in der AWS-Cloud verfügen.

Verwenden Sie diese Navigationstabelle, um schnell Antworten auf diese Fragen zu finden:

Wenn du willst.	Lesen.
Informieren Sie sich über die Kosten für den Betrieb dieser Lösung. Die geschätzten Kosten für den Betrieb dieser Lösung in der Region USA Ost (Nord-Virginia) belaufen sich auf USD 30,90 pro Monat für AWS-Ressourcen.	Kosten
Machen Sie sich mit den Sicherheitsüberlegungen für diese Lösung vertraut.	Sicherheit
Erfahren Sie, wie Sie Kontingente für diese Lösung einplanen.	Kontingente
Erfahren Sie, welche AWS-Regionen diese Lösung unterstützen.	Unterstützte AWS-Regionen
Erfahren Sie mehr über den optionalen MCP-Server für KI-gestützte Lasttestanalysen.	MCP-Serverintegration
Sehen Sie sich die in dieser Lösung enthaltene CloudFormation AWS-Vorlage an oder laden Sie sie herunter, um die Infrastrukturressourcen (den „Stack“) für diese Lösung automatisch bereitzustellen.	CloudFormation AWS-Vorlage
Greifen Sie auf den Quellcode zu und verwenden Sie optional das AWS Cloud Development Kit (AWS CDK), um die Lösung bereitzustellen.	GitHub Repository

Features

Die Lösung bietet die folgenden Funktionen:

Support mehrerer Test-Frameworks

Unterstützt JMeter K6- und Locust-Testskripte sowie einfache HTTP-Endpunkttests, ohne dass benutzerdefinierte Skripts erforderlich sind. Weitere Informationen finden Sie unter Testtypen im Abschnitt Architekturdetails.

Simulation mit hoher Benutzerauslastung

Simuliert Zehntausende gleichzeitiger virtueller Benutzer, um Ihre Anwendung unter realistischen Lastbedingungen einem Stresstest zu unterziehen.

Lastverteilung auf mehrere Regionen

Verteilt Belastungstests auf mehrere AWS-Regionen, um den geografisch verteilten Benutzerverkehr zu simulieren und die globale Leistung zu bewerten.

Flexible Testplanung

Plant Tests so, dass sie sofort, zu einem bestimmten future Datum und zu einer bestimmten Uhrzeit oder nach einem wiederkehrenden Zeitplan ausgeführt werden. Dabei werden Cron-Ausdrücke für automatisierte Regressionstests verwendet.

Überwachung in Echtzeit

Bietet optionales Live-Datenstreaming zur Überwachung des Testfortschritts mit Echtzeitmetriken wie Antwortzeiten, Anzahl virtueller Benutzer und Erfolgsquoten von Anfragen.

Umfassende Testergebnisse

Zeigt detaillierte Testergebnisse mit Leistungskennzahlen, Perzentilen (p50, p90, p95, p99), Fehleranalysen und herunterladbaren Artefakten für die Offline-Analyse an.

Vergleich der Ausgangswerte

Definiert Basis-Testläufe für den Leistungsvergleich, um Verbesserungen oder Regressionen im Laufe der Zeit nachzuverfolgen.

Flexibilität der Endgeräte

Testet jeden HTTP- oder HTTPS-Endpunkt in AWS-Regionen, lokalen Umgebungen oder anderen Cloud-Anbietern.

Intuitive Webkonsole

Bietet eine webbasierte Konsole zum Erstellen, Verwalten und Überwachen von Tests, ohne dass eine Befehlszeileninteraktion erforderlich ist.

KI-gestützte Analyse (optional)

Lässt sich über den Model Context Protocol (MCP) -Server in KI-Entwicklungstools integrieren und ermöglicht so eine intelligente Analyse von Lasttestdaten.

Support mehrerer Protokolle

Unterstützt verschiedene Protokolle wie HTTP, HTTPS, JDBC WebSocket, JMS, FTP und gRPC durch benutzerdefinierte Testskripte.

Vorteile

Die Lösung bietet die folgenden Vorteile:

Umfassende Leistungstests

Unterstützt Lasttests, Stresstests und Dauertests, um die Anwendungsleistung unter verschiedenen Bedingungen gründlich zu bewerten.

Früherkennung von Problemen

Identifiziert Leistungsengpässe, Speicherlecks und Skalierbarkeitsprobleme vor der Produktionsbereitstellung und reduziert so das Risiko von Ausfällen.

Simulation der Nutzung in der realen Welt

Simuliert genaues Benutzerverhalten und Verkehrsmuster in der realen Welt, um die Anwendungsleistung unter realistischen Bedingungen zu überprüfen.

Umsetzbare Performance Insights

Bietet detaillierte Metriken, Perzentile und Fehleranalysen, um das Anwendungsverhalten zu verstehen und Optimierungsmaßnahmen zu steuern.

Automatisierte Test-Workflows

Ermöglicht geplante und wiederkehrende Tests für kontinuierliche Leistungsüberwachung und Regressionstests ohne manuelles Eingreifen.

Kosteneffiziente Infrastruktur

Verwendet serverlose AWS Fargate-Container mit pay-per-use Preisgestaltung, sodass keine spezielle Testinfrastruktur und laufende Abonnementgebühren erforderlich sind.

Schnelle Testbereitstellung

Stellt die Testinfrastruktur innerhalb von Minuten bereit und skaliert sie, ohne Server bereitstellen oder verwalten zu müssen.

Einfache Abfrage der Testergebnisse

Lässt sich über einen optionalen Model Context Protocol (MCP) -Server in KI-Entwicklungstools integrieren und ermöglicht Abfragen in natürlicher Sprache und eine intelligente Analyse von Lasttestdaten für schnellere Einblicke und Problembehebung.

Anwendungsfälle

Validierung vor der Produktion

Testen Sie Web- und Mobilanwendungen unter produktionsähnlichen Lastbedingungen, bevor Sie eine neue Version starten, um die Leistung zu überprüfen und Probleme zu identifizieren.

Kapazitätsplanung

Ermitteln Sie die maximale Anzahl gleichzeitiger Benutzer, die Ihre Anwendung mit der aktuellen Infrastruktur unterstützen kann, und ermitteln Sie, wann eine Skalierung erforderlich ist.

Überprüfung bei Spitzenlast

Stellen Sie sicher, dass Ihre Infrastruktur Spitzenlasten, saisonale Verkehrsspitzen oder unerwartete Bedarfsspitzen ohne Leistungseinbußen bewältigen kann.

Leistungsoptimierung

Identifizieren Sie Leistungsengpässe wie langsame Datenbankabfragen, ineffiziente Codeausführung, Netzwerklatenz oder Ressourcenbeschränkungen.

Regressionstests

Planen Sie wiederkehrende Lasttests, um Leistungseinbußen zu erkennen, die durch neue Codebereitstellungen oder Infrastrukturänderungen verursacht werden.

Globale Leistungsbewertung

Bewerten Sie die Anwendungsleistung in mehreren geografischen Regionen, um ein einheitliches Benutzererlebnis für ein globales Publikum sicherzustellen.

API-Lasttests

Testen Sie REST APIs, GraphQL-Endpunkte oder Microservices, um Antwortzeiten, Durchsatz und Fehlerraten unter Last zu überprüfen.

Integration der CI/CD-Pipeline

Integrieren Sie automatisierte Leistungstests in kontinuierliche Integrations- und Bereitstellungspipelines, um Leistungsprobleme schon früh im Entwicklungszyklus zu erkennen.

Testen von Diensten durch Dritte

Testen Sie die Leistung und Zuverlässigkeit von Drittanbietern APIs oder Diensten, von denen Ihre Anwendung abhängt, unter verschiedenen Lastbedingungen.

Konzepte und Definitionen

In diesem Abschnitt werden die wichtigsten Konzepte beschrieben und die für diese Lösung spezifische Terminologie definiert:

Szenario

Testdefinition, einschließlich des Testnamens, der Beschreibung, der Anzahl der Aufgaben, der Parallelität, der AWS-Region, des Hochlaufs, der Wartezeit, des Testtyps, des geplanten Datums und der Wiederholungskonfigurationen.

Anzahl der Aufgaben

Anzahl der Container, die im Fargate-Cluster gestartet werden, um das Testszenario auszuführen. Zusätzliche Aufgaben werden nicht mehr erstellt, sobald das Kontolimit für Fargate-Ressourcen erreicht ist. Aufgaben, die bereits ausgeführt werden, werden jedoch fortgesetzt.

concurrency

Die Parallelität (Anzahl gleichzeitiger virtueller Benutzer pro Aufgabe). Basierend auf den Standardeinstellungen wird eine Parallelität von 200 empfohlen. Die Parallelität ist durch CPU und Arbeitsspeicher begrenzt. Bei Tests, die auf Apache basieren JMeter, erhöht eine höhere Parallelität den von der JVM für die ECS-Task genutzten Speicher. Die standardmäßige ECS-Aufgabendefinition erstellt Aufgaben mit 4 GB Arbeitsspeicher. Es wird empfohlen, mit niedrigeren Parallelitätswerten für eine Aufgabe zu beginnen und die CloudWatch ECS-Metriken für den Task-Cluster zu überwachen. Weitere Informationen finden Sie in den Kennzahlen zur Nutzung von Amazon ECS-Clustern.

Hochlauf

Der Zeitraum, in dem schrittweise von Null auf das angestrebte Parallelitätsniveau angehoben werden soll.

gedrückt halten für

Der Zeitraum, in dem das angestrebte Parallelitätsniveau nach Abschluss des Hochlaufs beibehalten werden soll.

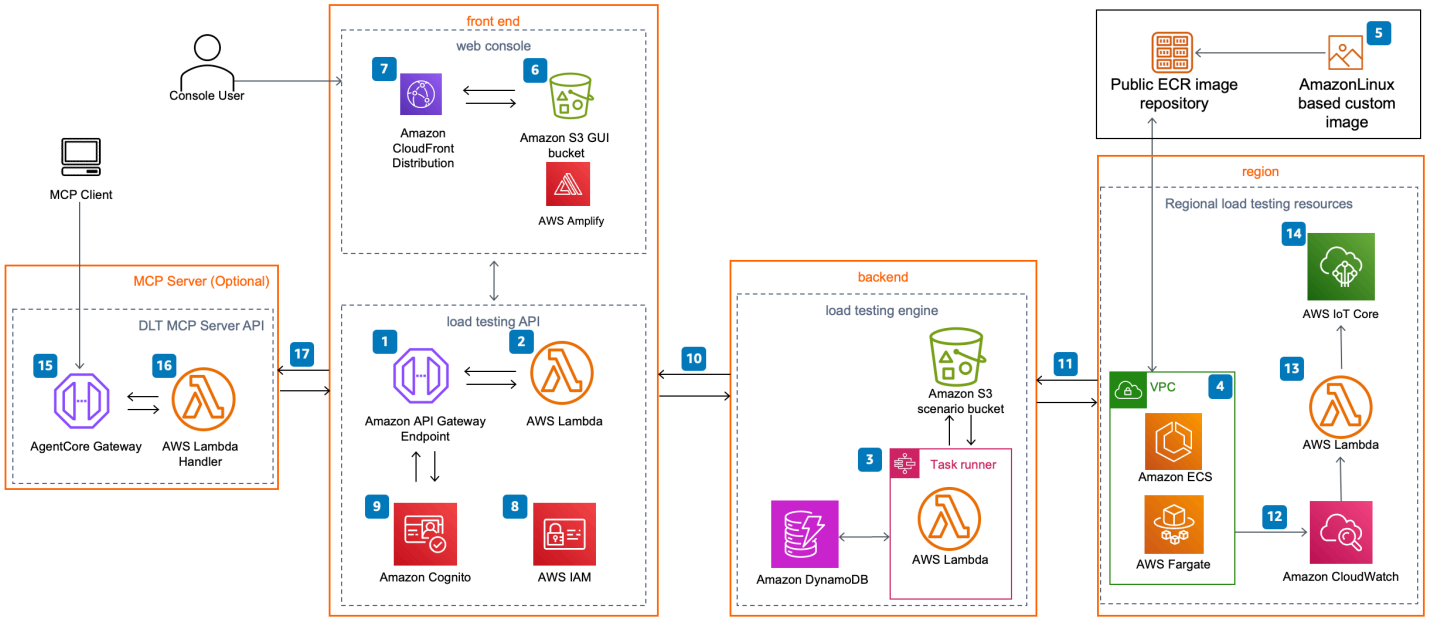
Eine allgemeine Referenz zu AWS-Begriffen finden Sie im AWS-Glossar.

Übersicht über die Architektur

Architekturdiagramm

Durch die Bereitstellung dieser Lösung mit den Standardparametern werden die folgenden Komponenten in Ihrem AWS-Konto bereitgestellt.

Verteilte Lasttests auf der AWS-Architektur auf AWS



> **Note**
>
> CloudFormation AWS-Ressourcen werden aus Konstrukten des AWS Cloud Development Kit (AWS CDK) erstellt.

Der allgemeine Prozessablauf für die mit der CloudFormation AWS-Vorlage bereitgestellten Lösungskomponenten sieht wie folgt aus:

1. Eine verteilte Loadtester-API nutzt Amazon API Gateway, um die Microservices der Lösung (AWS Lambda Lambda-Funktionen) aufzurufen.
2. Die Microservices stellen die Geschäftslogik zur Verwaltung von Testdaten und zum Ausführen der Tests bereit.

3. Diese Microservices interagieren mit Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB und AWS Step Functions, um Details und Ergebnisse von Testszenarien zu speichern und die Testausführung zu orchestrieren.
4. Eine Amazon Virtual Private Cloud (Amazon VPC) -Netzwerktopologie wird bereitgestellt, die die Amazon Elastic Container Service (Amazon ECS) -Container der Lösung enthält, die auf AWS Fargate ausgeführt werden.
5. Die Container verwenden ein Amazon Linux 2023-Basis-Image, auf dem das Taurus Load Testing Framework installiert ist. Taurus ist ein Open-Source-Framework zur Testautomatisierung, das K6 JMeter, Locust und andere Testtools unterstützt. Das Container-Image entspricht der Open Container Initiative (OCI) und wird von AWS in einem öffentlichen Repository von Amazon Elastic Container Registry (Amazon ECR) gehostet. Weitere Informationen finden Sie unter Anpassung des Container-Images.
6. Eine von AWS Amplify betriebene Webkonsole wird in einem S3-Bucket bereitgestellt, der für statisches Webhosting konfiguriert ist.
7. Amazon CloudFront bietet sicheren, öffentlichen Zugriff auf die Bucket-Inhalte der Website der Lösung.
8. Bei der Erstkonfiguration erstellt die Lösung eine standardmäßige Administratorrolle (IAM-Rolle) und sendet eine Zugangseinladung an eine vom Kunden angegebene Benutzer-E-Mail-Adresse.
9. Ein Amazon Cognito Cognito-Benutzerpool verwaltet den Benutzerzugriff auf die Konsole, die Distributed Load Tester API und den MCP-Server.
10. Nachdem Sie diese Lösung bereitgestellt haben, können Sie die Webkonsole verwenden oder APIs Testszenarien erstellen und ausführen, die eine Reihe von Aufgaben definieren.
11. Die Microservices verwenden dieses Testszenario, um ECS-Aufgaben auf Fargate in den angegebenen Regionen auszuführen.
12. Wenn der Test abgeschlossen ist, speichert die Lösung die Ergebnisse in S3 und DynamoDB und protokolliert die Ausgabe in Amazon. CloudWatch
13. Wenn Sie die Live-Datenoption aktivieren, sendet die Lösung während des Tests für jede Region, in der der Test ausgeführt wird, CloudWatch Protokolle von den Fargate-Aufgaben an eine Lambda-Funktion.
14. Die Lambda-Funktion veröffentlicht die Daten zum entsprechenden Thema in AWS IoT Core in der Region, in der der Haupt-Stack bereitgestellt wurde. Die Webkonsole abonniert das Thema und zeigt Echtzeitdaten an, während der Test ausgeführt wird.

> **Note**
>
> In den folgenden Schritten wird die optionale MCP-Serverintegration für KI-gestützte Lasttestanalysen beschrieben. Diese Komponente wird nur bereitgestellt, wenn Sie bei der Bereitstellung der Lösung die Option MCP-Server auswählen.

15. Ein MCP-Client (KI-Entwicklungstool) stellt eine Verbindung zum AgentCore AWS-Gateway-Endpunkt her, um über das Model Context Protocol auf die Daten der Distributed Load Testing-Lösung zuzugreifen. AgentCore Gateway validiert das Cognito-Authentifizierungstoken des Benutzers, um den autorisierten Zugriff auf den MCP-Server sicherzustellen.
16. Nach erfolgreicher Authentifizierung leitet AgentCore Gateway die MCP-Tool-Anfrage an die DLT MCP Server Lambda-Funktion weiter. Die Lambda-Funktion gibt die strukturierten Daten an AgentCore Gateway zurück, das sie für KI-gestützte Analysen und Einblicke an den MCP-Client zurücksendet.
17. Die Lambda-Funktion verarbeitet die Anfrage und fragt die entsprechenden AWS-Ressourcen (DynamoDB-Tabellen, S3-Buckets oder CloudWatch Logs) ab, um die angeforderten Lasttestdaten abzurufen.

Überlegungen zum AWS-Well-Architected-Design

Diese Lösung nutzt die Best Practices des AWS Well-Architected Framework, das Kunden dabei unterstützt, zuverlässige, sichere, effiziente und kostengünstige Workloads in der Cloud zu entwerfen und zu betreiben.

In diesem Abschnitt wird beschrieben, wie die Entwurfsprinzipien und Best Practices des Well-Architected Framework dieser Lösung zugute kommen.

Operative Exzellenz

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Verfahren des Pfeilers Operational Excellence konzipiert haben.

- Alle Ressourcen werden mithilfe von CloudFormation AWS-Vorlagen, die aus AWS-CDK-Konstrukten generiert wurden, als Infrastruktur als Code definiert.

- Die Lösung überträgt Metriken in verschiedenen CloudWatch Phasen, um die Beobachtbarkeit von Lambda-Funktionen, ECS-Aufgaben, S3-Buckets und anderen Lösungskomponenten zu gewährleisten.

Sicherheit

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Verfahren der Sicherheitssäule konzipiert haben.

- Cognito authentifiziert und autorisiert Webkonsolenbenutzer und API-Anfragen.
- Die gesamte dienstübergreifende Kommunikation verwendet AWS Identity and Access Management (IAM) -Rollen mit geringsten Zugriffsrechten, die nur die erforderlichen Mindestberechtigungen enthalten.
- Der gesamte Datenspeicher, einschließlich S3-Buckets und DynamoDB-Tabellen, verschlüsselt Daten im Ruhezustand mithilfe von AWS-verwalteten Schlüsseln.
- Protokollierung, Rückverfolgung und Versionierung werden gegebenenfalls zu Prüfungs- und Compliance-Zwecken aktiviert.
- Der Netzwerkzugriff ist standardmäßig privat, wobei VPC-Endpunkte aktiviert sind, sofern verfügbar, um den Datenverkehr innerhalb des AWS-Netzwerks aufrechtzuerhalten.

> **Note**
>
> Die Lösung erstellt mehrere CloudWatch Protokollgruppen mit unterschiedlichen Aufbewahrungszeiträumen, die auf dem Protokollvolumen und Kostenaspekten basieren:

Protokolltyp	Aufbewahrungszeitraum
Einblicke in ECS-Container	1 Tag
Step Functions, benutzerdefinierte ECS-Protokolle, API-Gateway-Zugriffsprotokolle	1 Jahr
Lambda-Laufzeitprotokolle	2 Jahre
API Gateway Gateway-Ausführung sprotokolle	Läuft niemals ab

Sie können diese Aufbewahrungsfristen in der CloudWatch Konsole Ihren Anforderungen entsprechend ändern.

Zuverlässigkeit

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Verfahren der Zuverlässigkeitskomponente konzipiert haben.

- Die Lösung verwendet, wo immer möglich, serverlose AWS-Services (Beispiele: Lambda, API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB und AWS Fargate), um eine hohe Verfügbarkeit und Wiederherstellung nach einem Serviceausfall sicherzustellen.
- Die gesamte Datenverarbeitung verwendet Lambda-Funktionen oder Amazon ECS auf AWS Fargate.
- Daten werden in DynamoDB und Amazon S3 gespeichert, sodass sie standardmäßig in mehreren Availability Zones gespeichert werden.

Leistungseffizienz

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Verfahren des Pfeilers Leistungseffizienz konzipiert haben.

- Die Lösung verwendet eine serverlose Architektur, die bei Bedarf horizontal skaliert werden kann.
- Die Lösung kann in jeder Region eingeführt werden, die die AWS-Services in dieser Lösung unterstützt, z. B.: AWS Lambda, Amazon API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB, Amazon ECS, AWS Fargate und Amazon Cognito.
- Die Lösung verwendet durchgehend Managed Services, um den betrieblichen Aufwand für die Bereitstellung und Verwaltung von Ressourcen zu reduzieren.
- Die Lösung wird täglich automatisch getestet und bereitgestellt, um Konsistenz zu gewährleisten, wenn sich die AWS-Services ändern. Außerdem wird sie von Lösungsarchitekten und Fachexperten auf Bereiche geprüft, in denen experimentierfreudig und verbesserungswürdig ist.

Kostenoptimierung

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Methoden der Säule Kostenoptimierung konzipiert haben.

- Die Lösung verwendet eine serverlose Architektur. Daher wird den Kunden nur das in Rechnung gestellt, was sie tatsächlich nutzen.
- Amazon DynamoDB skaliert die Kapazität nach Bedarf, sodass Sie nur für die Kapazität zahlen, die Sie tatsächlich nutzen.
- AWS ECS auf AWS Fargate ermöglicht es Ihnen, nur für die Rechenressourcen zu zahlen, die Sie nutzen, ohne Vorabkosten.
- AWS AgentCore Gateway dient als kostengünstiger Lambda-basierter Proxy für die API für verteilte Lasttests, wodurch die Notwendigkeit einer dedizierten Infrastruktur entfällt und die Kosten durch serverlose Preisgestaltung gesenkt werden. pay-per-request

Nachhaltigkeit

In diesem Abschnitt wird beschrieben, wie wir diese Lösung unter Verwendung der Prinzipien und bewährten Verfahren der Säule Nachhaltigkeit konzipiert haben.

- Die Lösung verwendet verwaltete serverlose Dienste, um die Umweltbelastung der Back-End-Dienste im Vergleich zu kontinuierlich betriebenen lokalen Diensten zu minimieren.
- Serverlose Dienste ermöglichen es Ihnen, nach Bedarf nach oben oder unten zu skalieren.

Einzelheiten zur Architektur

In diesem Abschnitt werden die Komponenten und AWS-Services beschrieben, aus denen diese Lösung besteht, sowie die Architekturdetails dazu, wie diese Komponenten zusammenarbeiten.

Die Lösung Distributed Load Testing on AWS besteht aus drei Komponenten auf hoher Ebene: einem Frontend, einem Backend und einem optionalen MCP-Server.

Frontend

Das Frontend bietet die Schnittstellen für die Interaktion mit der Lösung und umfasst:

- Eine Lasttest-API für den programmatischen Zugriff
- Eine Webkonsole zum Erstellen, Planen und Ausführen von Leistungstests
- Ein optionaler MCP-Server für die KI-gestützte Analyse von Testergebnissen und Fehlern

Belastungstest-API

Distributed Load Testing auf AWS konfiguriert Amazon API Gateway so, dass es die RESTful API der Lösung hostet. Benutzer können über die mitgelieferte Webkonsole, RESTful API und den optionalen MCP-Server sicher mit dem Lasttestsystem interagieren. Die API fungiert als „Eingangstür“ für den Zugriff auf Testdaten, die in Amazon DynamoDB gespeichert sind. Sie können die auch verwenden APIs , um auf alle erweiterten Funktionen zuzugreifen, die Sie in die Lösung integriert haben.

Diese Lösung nutzt die Benutzerauthentifizierungsfunktionen der Amazon Cognito Cognito-Benutzerpools. Nach erfolgreicher Authentifizierung eines Benutzers gibt Amazon Cognito ein JSON-Web-Token aus, mit dem die Konsole Anfragen an die Lösung APIs (Amazon API Gateway Gateway-Endpunkte) senden kann. HTTPS-Anfragen werden von der Konsole APIs mit dem Autorisierungsheader, der das Token enthält, an die Konsole gesendet.

Basierend auf der Anfrage ruft API Gateway die entsprechende AWS Lambda Lambda-Funktion auf, um die erforderlichen Aufgaben mit den in den DynamoDB-Tabellen gespeicherten Daten auszuführen, Testszenarien als JSON-Objekte in Amazon S3 zu speichern, Amazon CloudWatch Metrics-Bilder abzurufen und Testszenarien an die AWS Step Functions Functions-Zustandsmaschine zu senden.

Weitere Informationen zur API der Lösung finden Sie im Abschnitt Distributed Load Testing API in diesem Handbuch.

Web-Konsole

Diese Lösung umfasst eine Webkonsole, mit der Sie Tests konfigurieren und ausführen, laufende Tests überwachen und detaillierte Testergebnisse anzeigen können. Die Konsole ist eine ReactJS-Anwendung, die mit Cloudscape, einem Open-Source-Designsystem für die Erstellung intuitiver Webanwendungen, erstellt wurde. Die Konsole wird in Amazon S3 gehostet und der Zugriff erfolgt über Amazon CloudFront. Die Anwendung nutzt AWS Amplify zur Integration mit Amazon Cognito, um Benutzer zu authentifizieren. Die Webkonsole enthält auch eine Option zum Anzeigen von Live-Daten für einen laufenden Test, in dem sie das entsprechende Thema in AWS IoT Core abonniert.

Die URL der Webkonsole ist der Name der CloudFront Distributionsdomain, der in den CloudFormation Ausgaben als Konsole zu finden ist. Nachdem Sie die CloudFormation Vorlage gestartet haben, erhalten Sie außerdem eine E-Mail mit der URL der Webkonsole und dem Einmalkennwort für die Anmeldung.

MCP-Server (optional)

Der optionale Model Context Protocol (MCP) -Server bietet eine zusätzliche Schnittstelle für KI-Entwicklungstools, um über Interaktionen in natürlicher Sprache auf Lasttestdaten zuzugreifen und diese zu analysieren. Diese Komponente wird nur bereitgestellt, wenn Sie bei der Bereitstellung der Lösung die Option MCP-Server auswählen.

Mit dem MCP Server können KI-Agenten mithilfe von Tools wie Amazon Q, Claude und anderen MCP-kompatiblen KI-Assistenten Testergebnisse abfragen, Leistungskennzahlen analysieren und Einblicke in Ihre Lasttestdaten gewinnen. Ausführliche Informationen zur Architektur und Konfiguration des MCP-Servers finden Sie in diesem Abschnitt unter MCP Server.

Backend

Das Backend besteht aus einer Container-Image-Pipeline und einer Load-Test-Engine, mit der Sie die Last für die Tests generieren. Sie interagieren mit dem Backend über das Frontend. Darüber hinaus werden die für jeden Test gestarteten Aufgaben von Amazon ECS auf AWS Fargate mit einer eindeutigen Test-ID (ID) gekennzeichnet. Diese Test-ID-Tags können Ihnen helfen, die Kosten für diese Lösung zu überwachen. Weitere Informationen finden Sie unter Benutzerdefinierte Cost Allocation Tags im AWS Billing and Cost Management-Benutzerhandbuch.

Pipeline für Container-Images

Diese Lösung verwendet ein mit Amazon Linux 2023 erstelltes Container-Image als Basis-Image, auf dem das Taurus Load Testing Framework installiert ist. Taurus ist ein Open-Source-Framework zur Testautomatisierung, das K6 JMeter, Locust und andere Testtools unterstützt. AWS hostet dieses Image in einem öffentlichen Repository von Amazon Elastic Container Registry (Amazon ECR). Die Lösung verwendet dieses Image, um Aufgaben im Amazon ECS on AWS Fargate-Cluster auszuführen.

Weitere Informationen finden Sie im Abschnitt zur Anpassung von Container-Images in diesem Handbuch.

Infrastruktur testen

Zusätzlich zur CloudFormation Hauptvorlage bietet die Lösung eine regionale Vorlage, mit der die erforderlichen Ressourcen für die Durchführung von Tests in mehreren Regionen bereitgestellt werden können. Die Lösung speichert diese Vorlage in Amazon S3 und stellt in der Webkonsole einen Link dazu bereit. Jeder regionale Stack umfasst eine VPC, einen AWS Fargate-Cluster und eine Lambda-Funktion für die Verarbeitung von Live-Daten.

Weitere Informationen zur Bereitstellung der Testinfrastruktur in weiteren Regionen finden Sie im Abschnitt Bereitstellung in mehreren Regionen dieses Handbuchs.

Test-Engine auslasten

Die Distributed Load Testing-Lösung verwendet Amazon Elastic Container Service (Amazon ECS) und AWS Fargate, um Tausende von gleichzeitigen Benutzern in mehreren Regionen zu simulieren und HTTP-Anfragen mit kontinuierlicher Geschwindigkeit zu generieren.

Sie definieren die Testparameter mithilfe der mitgelieferten Webkonsole. Die Lösung verwendet diese Parameter, um ein JSON-Testszenario zu generieren und es in Amazon S3 zu speichern. Weitere Informationen zu Testskripten und Testparametern finden Sie unter Testtypen in diesem Abschnitt.

Eine AWS Step Functions Functions-Zustandsmaschine führt Amazon ECS-Aufgaben in einem AWS Fargate-Cluster aus und überwacht sie. Die AWS Step Functions Functions-Zustandsmaschine umfasst eine AWS-Lambda-Funktion mit ecr-checker, eine AWS-Lambda-Funktion, eine task-status-checker AWS-Lambda-Funktion für Task-Runner, eine AWS-Lambda-Funktion zum Abbrechen von Aufgaben und eine AWS-Lambda-Funktion zum Analysieren von Ergebnissen. Weitere Informationen zum Workflow finden Sie im Abschnitt „Workflow testen“ dieses Handbuchs. Weitere Informationen zu

Testergebnissen finden Sie im Abschnitt Testergebnisse dieses Handbuchs. Weitere Informationen zum Ablauf der Teststornierung finden Sie im Abschnitt zum Ablauf der Teststornierung in diesem Handbuch.

Wenn Sie Live-Daten auswählen, initiiert die Lösung in jeder Region eine real-time-data-publisher Lambda-Funktion anhand der CloudWatch Protokolle, die den Fargate-Aufgaben in dieser Region entsprechen. Die Lösung verarbeitet und veröffentlicht dann die Daten zu einem Thema in AWS IoT Core in der Region, in der Sie den Haupt-Stack gestartet haben. Weitere Informationen finden Sie im Abschnitt Live-Daten dieses Handbuchs.

MCP-Server

Die optionale Serverintegration des Model Context Protocol (MCP) ermöglicht es KI-Agenten, über Interaktionen in natürlicher Sprache programmgesteuert auf Ihre Lasttestdaten zuzugreifen und diese zu analysieren. Diese Komponente wird nur bereitgestellt, wenn Sie bei der Bereitstellung der Lösung die Option MCP-Server auswählen.

Der MCP-Server fungiert als Brücke zwischen KI-Entwicklungstools und Ihrer DLT-Bereitstellung und bietet eine standardisierte Schnittstelle für die intelligente Analyse der Ergebnisse von Leistungstests. Die Architektur integriert mehrere AWS-Services, um eine sichere, skalierbare Schnittstelle für Interaktionen mit KI-Agenten zu schaffen:

AgentCore AWS-Gateway

AWS AgentCore Gateway ist ein vollständig verwalteter Service, der standardisiertes Hosting und Protokollmanagement für MCP-Server bietet. In dieser Lösung dient AgentCore Gateway als öffentlicher Endpunkt, mit dem KI-Agenten eine Verbindung herstellen, wenn sie Zugriff auf Ihre Lasttestdaten anfordern.

Der Service wickelt die gesamte MCP-Protokollkommunikation ab, einschließlich der Erkennung von Tools, der Validierung von Authentifizierungstoken und der Weiterleitung von Anfragen. AgentCore Gateway arbeitet als Mehrmandantendienst mit integrierten Sicherheitsvorkehrungen gegen allgemeine Bedrohungen für öffentliche Endgeräte und validiert gleichzeitig die Signaturen und Ansprüche von Cognito-Tokens für jede Anfrage.

DLT MCP Server Lambda

Die DLT MCP Server Lambda-Funktion ist eine benutzerdefinierte serverlose Komponente, die MCP-Anfragen von AI-Agenten verarbeitet und sie in Abfragen für Ihre DLT-Ressourcen übersetzt.

Diese Lambda-Funktion fungiert als Informationsebene der MCP-Integration. Sie ruft Testergebnisse aus DynamoDB-Tabellen ab, greift auf Leistungsartefakte zu, die in S3-Buckets gespeichert sind, und fragt CloudWatch Logs nach detaillierten Ausführungsinformationen ab. Die Lambda-Funktion implementiert schreibgeschützte Zugriffsmuster und wandelt DLT-Rohdaten in strukturierte, KI-freundliche Formate um, die Agenten einfach interpretieren und analysieren können.

Integration der Authentifizierung

Das Authentifizierungssystem nutzt Ihre bestehende Cognito-Benutzerpool-Infrastruktur, um konsistente Zugriffskontrollen sowohl für die Webkonsole als auch für die MCP-Serverschnittstellen aufrechtzuerhalten.

Diese Integration verwendet die tokenbasierte OAuth 2.0-Authentifizierung. Benutzer authentifizieren sich einmal über den Cognito-Anmeldevorgang und erhalten Token, die sowohl für Benutzeroberflächeninteraktionen als auch für den MCP-Serverzugriff funktionieren. Das System behält dieselben Berechtigungsgrenzen und Zugriffskontrollen wie die Weboberfläche bei und stellt so sicher, dass Benutzer nur über KI-Agenten auf dieselben Lasttestdaten zugreifen können, auf die sie über die Konsole zugreifen können.

AWS-Services in dieser Lösung

Die folgenden AWS-Services sind in dieser Lösung enthalten:

AWS Service	Description
Amazon API Gateway	Kern. Hostet REST-API-Endpunkte in der Lösung.
AWS CloudFormation	Kern. Verwaltet Bereitstellungen für die Lösungsinfrastruktur.
Amazon CloudFront	Kern. Stellt die in Amazon S3 gehosteten Webinhalte bereit.
Amazon CloudWatch	Kern. Speichert die Lösungsprotokolle und Metriken.
Amazon Cognito	Kern. Verwaltet die Benutzerverwaltung und Authentifizierung für die API.
Amazon-DynamoDB	Kern. Speichert Bereitstellungsinformationen und testet Szenariodetails und Ergebnisse.

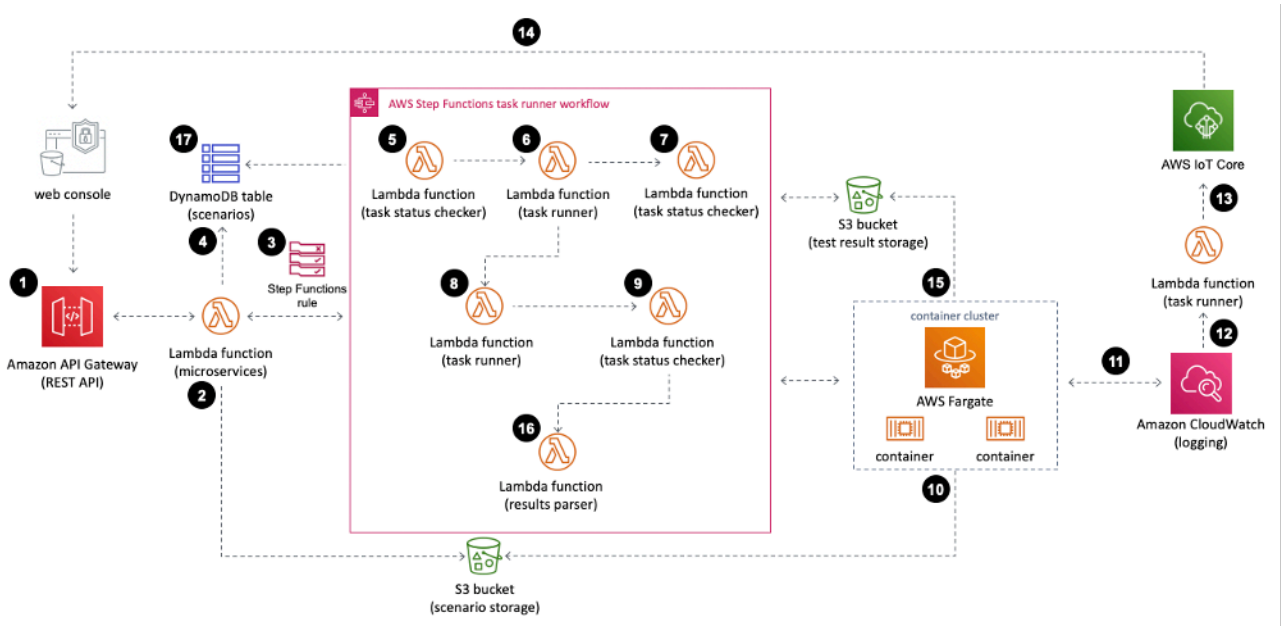
AWS Service	Description
Amazon Elastic Container Service	Kern. Stellt unabhängige Amazon ECS-Aufgaben auf AWS Fargate-Containern bereit und verwaltet sie.
AWS Fargate	Kern. Hostet die Amazon ECS-Container der Lösung
AWS Identity and Access Management	Kern. Kümmert sich um die Verwaltung von Benutzerrollen und Berechtigungen.
AWS Lambda	Kern. Stellt Logik für die APIs Implementierung, das Analysieren von Testergebnissen und das Starten von workers/leader Aufgaben bereit.
AWS Step Functions	Kern. Orchestriert die Bereitstellung von Amazon ECS-Containern für AWS Fargate-Aufgaben in den angegebenen Regionen
AWS Amplify	Unterstützend. Stellt eine von AWS Amplify betriebene Webkonsole bereit.
CloudWatch Amazon-Veranstaltungen	Unterstützend. Plant Tests so, dass sie automatisch an einem bestimmten Datum oder an wiederkehrenden Terminen beginnen.
Amazon Elastic Container Registry	Unterstützend. Hostet das Container-Image in einem öffentlichen ECR-Repository.
AWS IoT Core	Unterstützend. Ermöglicht die Anzeige von Live-Daten für einen laufenden Test, indem Sie das entsprechende Thema in AWS IoT Core abonnieren.
AWS Systems Manager	Unterstützend. Ermöglicht die Überwachung von Ressourcen auf Anwendungsebene und die Visualisierung von Ressourcenoperationen und Kostendaten.
Amazon S3	Unterstützend. Hostet die statischen Webinhalte, Protokolle, Metriken und Testdaten.
Amazon Virtual Private Cloud	Unterstützend. Enthält die Amazon ECS-Container der Lösung, die auf AWS Fargate ausgeführt werden.

AWS Service	Description
Amazon Bedrock AgentCore	Unterstützend, optional. Hostet den optionalen Remote Model Context Protocol (MCP) -Server der Lösung für die Integration von KI-Agenten mit der API.

So funktioniert Distributed Load Testing auf AWS

Die folgende detaillierte Aufschlüsselung zeigt die Schritte, die zur Ausführung eines Testszenarios erforderlich sind.

Test-Arbeitsablauf



1. Sie verwenden die Webkonsole, um ein Testszenario mit den Konfigurationsdetails an die API der Lösung zu senden.
2. Die Konfiguration des Testszenarios wird als JSON-Datei () in den Amazon Simple Storage Service (Amazon S3`s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json`) hochgeladen.
3. Ein AWS Step Functions Functions-Zustandsmaschine wird mit der Test-ID, der Anzahl der Aufgaben, dem Testtyp und dem Dateityp als Eingabe für die AWS Step Functions Functions-Zustandsmaschine ausgeführt. Wenn der Test geplant ist, erstellt er zunächst eine CloudWatch

Ereignisregel, die AWS Step Functions am angegebenen Datum auslöst. Weitere Informationen zum Planungs-Workflow finden Sie im Abschnitt Test-Scheduling-Workflow in diesem Handbuch.

4. Konfigurationsdetails werden in der Amazon DynamoDB-Szenario-Tabelle gespeichert.
5. Im Task Runner-Workflow von AWS Step Functions prüft die task-status-checker AWS Lambda Lambda-Funktion, ob Amazon Elastic Container Service (Amazon ECS) -Aufgaben bereits für dieselbe Test-ID ausgeführt werden. Wenn festgestellt wird, dass Aufgaben mit derselben Test-ID ausgeführt werden, führt dies zu einem Fehler. Wenn im AWS Fargate-Cluster keine Amazon ECS-Aufgaben ausgeführt werden, gibt die Funktion die Test-ID, die Anzahl der Aufgaben und den Testtyp zurück.
6. Die Task-Runner-Funktion AWS Lambda ruft die Aufgabendetails aus dem vorherigen Schritt ab und führt die Amazon ECS-Worker-Aufgaben im AWS Fargate-Cluster aus. Die Amazon ECS-API verwendet die RunTask Aktion, um die Worker-Aufgaben auszuführen. Diese Worker-Aufgaben werden gestartet und warten dann auf eine Startnachricht von der Leader-Aufgabe, um mit dem Test zu beginnen. Die RunTask Aktion ist auf 10 Aufgaben pro Definition begrenzt. Wenn die Anzahl Ihrer Aufgaben mehr als 10 beträgt, wird die Aufgabendefinition mehrmals ausgeführt, bis alle Worker-Aufgaben gestartet wurden. Die Funktion generiert auch ein Präfix, um den aktuellen Test in der AWS Lambda Lambda-Funktion zur Ergebnisanalyse zu unterscheiden.
7. Die task-status-checker AWS Lambda Lambda-Funktion prüft, ob alle Amazon ECS-Worker-Aufgaben mit derselben Test-ID ausgeführt werden. Wenn die Aufgaben noch bereitgestellt werden, wartet sie eine Minute und prüft erneut. Sobald alle Amazon ECS-Aufgaben ausgeführt werden, gibt es die Test-ID, die Anzahl der Aufgaben, den Testtyp, alle Aufgaben und das Präfix zurück IDs und übergibt sie an die Task-Runner-Funktion.
8. Die Task-Runner-Funktion AWS Lambda wird erneut ausgeführt. Diesmal wird eine einzelne Amazon ECS-Task gestartet, die als Leader-Knoten fungiert. Diese ECS-Aufgabe sendet eine Starttestnachricht an jede der Worker-Aufgaben, damit die Tests gleichzeitig gestartet werden können.
9. Die task-status-checker AWS Lambda Lambda-Funktion überprüft erneut, ob Amazon ECS-Aufgaben mit derselben Test-ID ausgeführt werden. Wenn Aufgaben noch ausgeführt werden, wartet sie eine Minute und prüft erneut. Sobald keine Amazon ECS-Aufgaben ausgeführt werden, werden die Test-ID, die Anzahl der Aufgaben, der Testtyp und das Präfix zurückgegeben.
10. Wenn die Task-Runner-Funktion AWS Lambda die Amazon ECS-Aufgaben im AWS Fargate-Cluster ausführt, lädt jede Aufgabe die Testkonfiguration von Amazon S3 herunter und startet den Test.
11. Sobald die Tests ausgeführt werden, werden die durchschnittliche Antwortzeit, die Anzahl der gleichzeitigen Benutzer, die Anzahl der erfolgreichen Anfragen und die Anzahl der

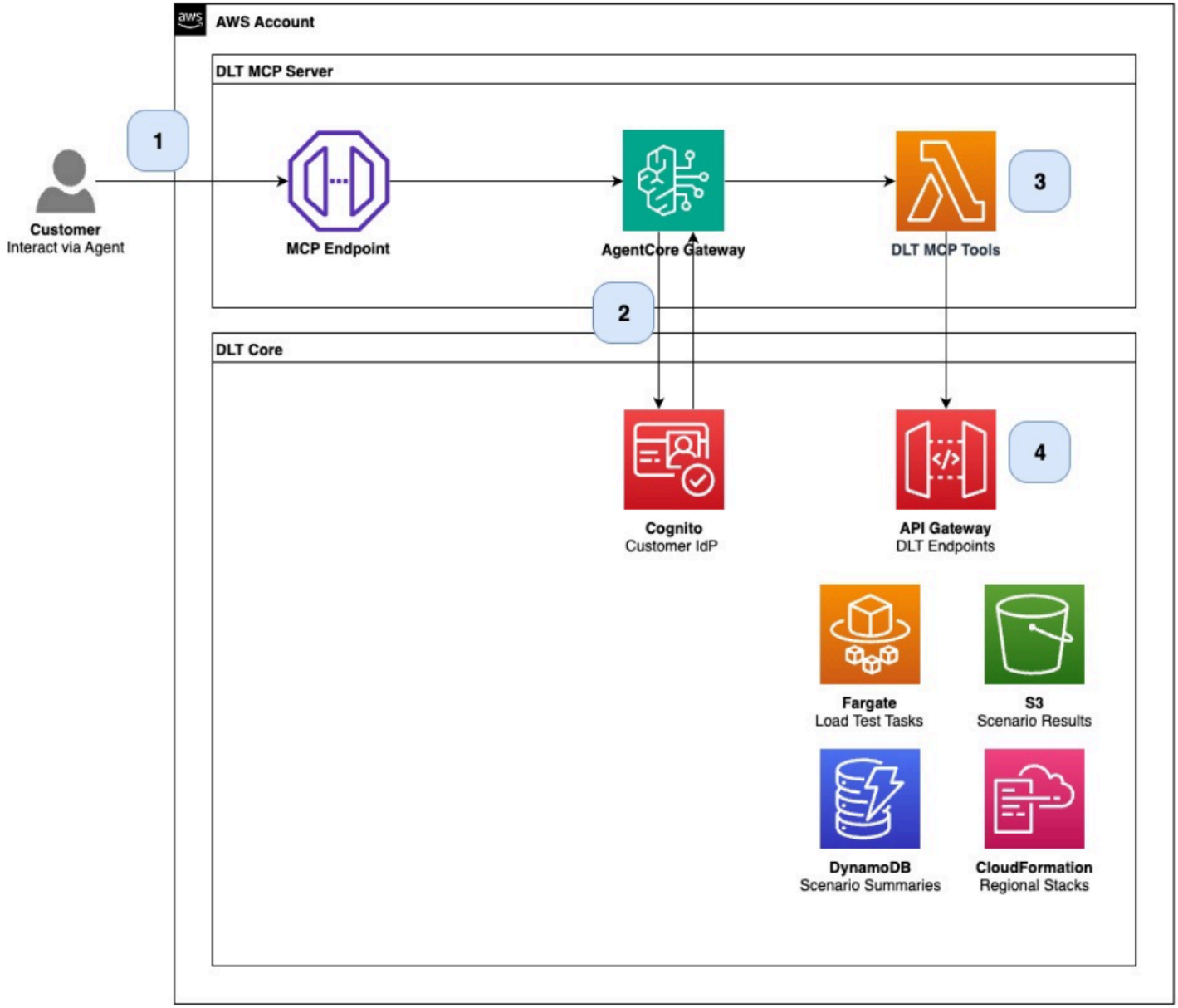
fehlgeschlagenen Anfragen für jede Aufgabe in Amazon protokolliert CloudWatch und können in einem CloudWatch Dashboard eingesehen werden.

12. Wenn Sie Live-Daten in den Test aufgenommen haben, filtert die Lösung Echtzeit-Testergebnisse CloudWatch mithilfe eines Abonnementfilters. Dann übergibt die Lösung die Daten an eine Lambda-Funktion.
13. Die Lambda-Funktion strukturiert dann die empfangenen Daten und veröffentlicht sie in einem AWS IoT Core Core-Thema.
14. Die Webkonsole abonniert das AWS IoT Core Core-Thema für den Test und empfängt die zum Thema veröffentlichten Daten, um die Echtzeitdaten während der Ausführung des Tests grafisch darzustellen.
15. Wenn der Test abgeschlossen ist, exportieren die Container-Images einen detaillierten Bericht als XML-Datei nach Amazon S3. Jede Datei erhält eine UUID für den Dateinamen. Zum Beispiel s3://dlte-bucket/test-scenarios/ <$TEST_ID> /results/ <$UUID> .json.
16. Wenn die XML-Dateien auf Amazon S3 hochgeladen werden, liest die AWS Lambda Lambda-Funktion für Ergebnisanalyse die Ergebnisse in den XML-Dateien, beginnend mit dem Präfix, analysiert und aggregiert alle Ergebnisse zu einem zusammengefassten Ergebnis.
17. Die AWS Lambda Lambda-Funktion für Ergebnisanalyse schreibt das Gesamtergebnis in eine Amazon DynamoDB-Tabelle.

MCP-Server-Workflow (optional)

Wenn Sie die optionale MCP-Serverintegration einsetzen, können KI-Agenten über den folgenden Workflow auf Ihre Lasttestdaten zugreifen und diese analysieren:

MCP-Serverarchitektur



1. Kundeninteraktion — Der Kunde interagiert mit dem MCP von DLT über den MCP-Endpunkt, der von AWS Gateway gehostet wird. AgentCore KI-Agenten stellen eine Verbindung zu diesem Endpunkt her, um Zugriff auf Belastungstestdaten anzufordern.
2. Autorisierung — AgentCore Gateway übernimmt die Autorisierung für den Solution Cognito-Benutzerpool-Anwendungsclient. Das Gateway validiert das Cognito-Token des Benutzers, um sicherzustellen, dass er berechtigt ist, auf den DLT-MCP-Server zuzugreifen. Autorisierten Benutzern wird Zugriff gewährt, wobei der Zugriff auf das Agententool auf schreibgeschützte Operationen beschränkt ist.

3. Toolspezifikation — AgentCore Gateway stellt eine Verbindung zur DLT MCP Server Lambda-Funktion her. Eine Toolspezifikation definiert die verfügbaren Tools, mit denen KI-Agenten mit Ihren Lasttestdaten interagieren können.
4. Schreibgeschützter API-Zugriff — Die Lambda-Funktion ist auf den schreibgeschützten API-Zugriff über die vorhandenen DLT API Gateway Gateway-Endpunkte beschränkt. Die Funktion bietet vier Hauptoperationen:
   - Szenarien auflisten — Ruft eine Liste von Testszenarien aus der DynamoDB-Szenariatabelle ab
   - Szenario-Testergebnisse abrufen — Greifen Sie auf detaillierte Testergebnisse für bestimmte Szenarien von DynamoDB und S3 zu
   - Holen Sie sich Fargate-Load-Test-Runner — Informationen zur Ausführung von Fargate-Aufgaben im ECS-Cluster abfragen
   - Verfügbare regionale Stacks abrufen — Rufen Sie Informationen über die bereitgestellte regionale Infrastruktur ab von CloudFormation

Die MCP-Serverintegration nutzt die bestehende DLT-Infrastruktur (API Gateway, Cognito, DynamoDB, S3), um sicheren, schreibgeschützten Zugriff auf Testdaten für KI-gestützte Analysen und Erkenntnisse zu ermöglichen.

Designüberlegungen

In diesem Abschnitt werden wichtige Designentscheidungen und Konfigurationsoptionen für die Distributed Load Testing on AWS-Lösung beschrieben, einschließlich unterstützter Anwendungen, Testtypen, Planungsoptionen und Überlegungen zur Bereitstellung.

Unterstützte Anwendungen

Diese Lösung unterstützt das Testen von cloudbasierten Anwendungen und lokalen Anwendungen, sofern Sie über eine Netzwerkverbindung zwischen Ihrem AWS-Konto und Ihrer Anwendung verfügen. Die Lösung unterstützt APIs die Verwendung von HTTP- oder HTTPS-Protokollen.

Testtypen

Distributed Load Testing auf AWS unterstützt mehrere Testtypen: einfache HTTP-Endpunkttests JMeter, K6 und Locust.

Einfache HTTP-Endpunkttests

Die Webkonsole bietet eine Schnittstelle zur Konfiguration von HTTP-Endpunkten, mit der Sie jeden HTTP- oder HTTPS-Endpunkt testen können, ohne benutzerdefinierte Skripts schreiben zu müssen. Sie definieren die Endpunkt-URL, wählen die HTTP-Methode (GET, POST, PUT, DELETE usw.) aus einem Dropdownmenü aus und fügen optional benutzerdefinierte Anforderungsheader und Body-Payloads hinzu. Diese Konfiguration ermöglicht es Ihnen, APIs mit benutzerdefinierten Autorisierungstoken, Inhaltstypen oder anderen HTTP-Headern und Anforderungstexten zu testen, die für Ihre Anwendung erforderlich sind.

JMeter Tests

Wenn Sie ein Testszenario mit der Webkonsole erstellen, können Sie ein JMeter Testskript hochladen. Die Lösung lädt das Skript in den S3-Bucket des Szenarios hoch. Wenn Amazon ECS-Aufgaben ausgeführt werden, laden sie das JMeter Skript von S3 herunter und führen den Test aus.

> **Important**
>
> Ihr JMeter Skript kann zwar Parallelität (virtuelle Benutzer), Transaktionsraten (TPS), Anlaufzeiten und andere Ladeparameter definieren, aber die Lösung überschreibt diese Konfigurationen mit den Werten, die Sie während der Testerstellung auf dem Traffic Shape-Bildschirm angeben. Die Traffic Shape-Konfiguration steuert die Anzahl der Aufgaben, die Parallelität (virtuelle Benutzer pro Aufgabe), die Dauer des Hochlaufs und die Haltedauer für die Testausführung.

Wenn Sie über JMeter Eingabedateien verfügen, können Sie die Eingabedateien zusammen mit dem Skript komprimieren. JMeter Sie können die ZIP-Datei auswählen, wenn Sie ein Testszenario erstellen.

Wenn Sie Plugins einbeziehen möchten, werden alle .jar-Dateien, die in einem /plugins-Unterverzeichnis der mitgelieferten ZIP-Datei enthalten sind, in das JMeter Erweiterungsverzeichnis kopiert und stehen dort für Auslastungstests zur Verfügung.

> **Note**
>
> Wenn Sie Ihrer JMeter Skriptdatei JMeter Eingabedateien hinzufügen, müssen Sie den relativen Pfad der Eingabedateien in Ihrer Skriptdatei angeben. JMeter Außerdem müssen sich die Eingabedateien im relativen Pfad befinden. Wenn sich Ihre JMeter Eingabedateien

und die Skriptdatei beispielsweise stattdessen im Verzeichnis/home/user directory and you refer to the input files in the JMeter script file, the path of input files must be ./INPUT_FILES. If you use /home/user/INPUT_FILES befinden, schlägt der Test fehl, da die Eingabedateien nicht gefunden werden können.

Wenn Sie JMeter Plugins einbeziehen, müssen die .jar-Dateien in einem Unterverzeichnis namens / plugins im Stammverzeichnis der ZIP-Datei gebündelt werden. Relativ zum Stammverzeichnis der ZIP-Datei muss der Pfad zu den JAR-Dateien sein. /Plugins/Bundled_Plugin.jar.

Weitere Informationen zur Verwendung von Skripten finden Sie im Benutzerhandbuch. JMeter JMeter

K6-Tests

Die Lösung unterstützt Tests auf Basis des K6-Frameworks. K6 ist unter der AGPL-3.0-Lizenz veröffentlicht. Die Lösung zeigt bei der Erstellung eines neuen K6-Tests eine Lizenzbestätigungsmeldung an. Sie können die K6-Testdatei zusammen mit allen erforderlichen Eingabedateien in eine Archivdatei hochladen.

Important

Ihr K6-Skript kann zwar Parallelität (virtuelle Benutzer), Stufen, Schwellenwerte und andere Ladeparameter definieren, aber die Lösung überschreibt diese Konfigurationen mit den Werten, die Sie bei der Testerstellung im Traffic Shape-Bildschirm angeben. Die Traffic Shape-Konfiguration steuert die Anzahl der Aufgaben, die Parallelität (virtuelle Benutzer pro Aufgabe), die Dauer des Hochlaufs und die Haltedauer für die Testausführung.

Locust testet

Die Lösung unterstützt Tests auf Basis des Locust Frameworks. Sie können die Locust-Testdatei zusammen mit allen erforderlichen Eingabedateien in eine Archivdatei hochladen.

Important

Ihr Locust-Skript kann zwar Parallelität (Benutzeranzahl), Spawn-Rate und andere Ladeparameter definieren, aber die Lösung überschreibt diese Konfigurationen mit den Werten, die Sie während der Testerstellung im Traffic Shape-Bildschirm angeben. Die Traffic

Shape-Konfiguration steuert die Anzahl der Aufgaben, die Parallelität (virtuelle Benutzer pro Aufgabe), die Dauer des Hochlaufs und die Haltedauer für die Testausführung.

Planung von Tests

Die Lösung bietet drei Optionen zum Ausführungszeitpunkt für die Ausführung von Lasttests:

- Jetzt ausführen — Führt den Belastungstest unmittelbar nach der Erstellung aus
- Einmal ausführen — Führen Sie den Test an einem bestimmten Datum und zu einer bestimmten Uhrzeit in der future aus
- Nach einem Zeitplan ausführen — Erstellen Sie wiederkehrende Tests mithilfe von Cron-Ausdrücken, um den Zeitplan zu definieren

Wenn Sie Einmal ausführen auswählen, geben Sie die Laufzeit im 24-Stunden-Format und das Ausführungsdatum an, an dem der Belastungstest ausgeführt werden soll.

Wenn Sie Nach Zeitplan ausführen auswählen, können Sie entweder manuell einen Cron-Ausdruck eingeben oder aus gängigen Cron-Mustern wählen (z. B. stündlich, täglich zu einer bestimmten Zeit, wochentags oder monatlich). Der Cron-Ausdruck verwendet ein detailliertes Zeitplanformat mit Feldern für Minuten, Stunden, Monatstag, Monat, Wochentag und Jahr. Sie müssen auch ein Ablaufdatum angeben, das festlegt, wann der geplante Test nicht mehr ausgeführt werden soll. Weitere Informationen zur Funktionsweise der Planung finden Sie im Abschnitt zum Ablauf der Testplanung in diesem Handbuch.

Note

- Testdauer: Berücksichtigen Sie bei der Planung die Gesamtdauer der Tests. Ein Test mit einer Anlaufzeit von 10 Minuten und einer Haltezeit von 40 Minuten dauert beispielsweise etwa 80 Minuten.
- Mindestintervall: Stellen Sie sicher, dass das Intervall zwischen den geplanten Tests länger ist als die geschätzte Testdauer. Wenn der Test beispielsweise etwa 80 Minuten dauert, sollten Sie ihn so planen, dass er nicht häufiger als alle 3 Stunden ausgeführt wird.
- Stündliche Begrenzung: Das System erlaubt es nicht, Tests mit einem Unterschied von nur einer Stunde zu planen, selbst wenn die geschätzte Testdauer weniger als eine Stunde beträgt.

Gleichzeitige Tests

Diese Lösung erstellt für jeden Test ein CloudWatch Amazon-Dashboard, das die kombinierte Ausgabe aller im Amazon ECS-Cluster ausgeführten Aufgaben in Echtzeit anzeigt. Das CloudWatch Dashboard zeigt die durchschnittliche Antwortzeit, die Anzahl der gleichzeitigen Benutzer, die Anzahl der erfolgreichen Anfragen und die Anzahl der fehlgeschlagenen Anfragen. Die Lösung aggregiert jede Metrik sekundengenau und aktualisiert das Dashboard jede Minute.

Benutzerverwaltung

Bei der Erstkonfiguration geben Sie einen Benutzernamen und eine E-Mail-Adresse an, die Amazon Cognito verwendet, um Ihnen Zugriff auf die Webkonsole der Lösung zu gewähren. Die Konsole bietet keine Benutzerverwaltung. Um weitere Benutzer hinzuzufügen, müssen Sie die Amazon Cognito Cognito-Konsole verwenden. Weitere Informationen finden Sie unter Managing Users in User Pools im Amazon Cognito Developer Guide.

Informationen zur Migration vorhandener Benutzer zu Amazon Cognito-Benutzerpools finden Sie im AWS-Blog Approaches for migration users to Amazon Cognito user pools.

Regionale Bereitstellung

Diese Lösung verwendet Amazon Cognito, das nur in bestimmten AWS-Regionen verfügbar ist. Daher müssen Sie diese Lösung in einer Region bereitstellen, in der Amazon Cognito verfügbar ist. Die aktuelle Serviceverfügbarkeit nach Regionen finden Sie in der regionalen AWS-Serviceliste.

Planen Sie Ihren Einsatz

In diesem Abschnitt werden Kosten, Sicherheit, unterstützte Regionen, Kontingente und andere Überlegungen beschrieben, die Sie vor der Bereitstellung der Lösung überprüfen sollten.

Cost (Kosten)

Sie sind für die Kosten der AWS-Services verantwortlich, die Sie beim Betrieb dieser Lösung in Anspruch nehmen. Die Gesamtkosten hängen von der Anzahl der ausgeführten Lasttests, der Dauer dieser Tests und der Menge der generierten Daten ab. Zum jetzigen Zeitpunkt belaufen sich die geschätzten Kosten für die Ausführung dieser Lösung mit Standardeinstellungen in der Region USA Ost (Nord-Virginia) auf etwa 30,90$ pro Monat.

Die folgende Tabelle enthält ein Beispiel für eine Aufschlüsselung der Kosten für die Bereitstellung dieser Lösung mit den Standardparametern in der Region USA Ost (Nord-Virginia) für einen Monat.

AWS Service	Dimensionen	Kosten [USD]
AWS Fargate	10 On-Demand-Aufgaben (mit zwei V CPUs - und 4 GB-Speicher), die 30 Stunden lang ausgeführt werden	29,62$
Amazon DynamoDB	1.000 On-Demand-Schreibkapazitätseinheiten 1.000 On-Demand-Lesekapazitätseinheiten	0,0015$
AWS Lambda	1.000 Anfragen Gesamtdauer 10 Minuten	1,25$
AWS Step Functions	1.000 Zustandsübergänge	0,025 USD
Insgesamt:		30,90$ pro Monat

Die Lösungsressourcen sind mit Key= und Value= SolutionId gekennzeichnet. SO0062 Sie können den Tag-Schlüssel aktivieren, SolutionId indem Sie der Dokumentation activating-tags folgen. Sobald das Tag aktiviert ist, können Sie eine Kostenkategorienregel erstellen, indem Sie der Dokumentation zur Erstellung von Kostenkategorien folgen. Sie können die für die Lösung angefallenen Kosten einsehen, indem Sie die Kostenkategorienkonsole überwachen und den Namen der Kostenkategorie auswählen.

Wir empfehlen, über den AWS Cost Explorer ein Budget zu erstellen, um die Kosten besser verwalten zu können. Die Preise sind freibleibend. Vollständige Informationen finden Sie auf der Preisseite für jeden AWS-Service, der in dieser Lösung verwendet wird.

> **Note**
>
> Die standardmäßige Aufgabenkonfiguration verwendet 2 V CPUs und 4 GB Arbeitsspeicher pro Aufgabe. Wenn Ihre Auslastungstests diese Ressourcen nicht benötigen, können Sie sie reduzieren, um die Kosten zu senken. Umgekehrt können Sie die Ressourcen erhöhen, um eine höhere Parallelität pro Aufgabe zu unterstützen. Weitere Informationen finden Sie im Abschnitt Erhöhen der Anzahl der Container-Ressourcen in diesem Handbuch.

> **Note**
>
> Diese Lösung bietet die Möglichkeit, Live-Daten bei der Ausführung eines Tests einzubeziehen. Für diese Funktion sind eine zusätzliche AWS Lambda Lambda-Funktion und ein AWS IoT Core Core-Thema erforderlich, für die zusätzliche Kosten anfallen.

Zusätzliche Kosten für MCP-Server (optional)

Die folgende Tabelle enthält eine Aufschlüsselung der Kosten für die MCP-Serverintegration mit Preisen in der Region USA Ost (Nord-Virginia) für einen Monat.

Servicekomponente	Dimensionen	Kosten [USD]
AgentCore Gateway — Indizierung von Tools	10 Werkzeuge × 0,02$ pro 100 Werkzeuge	0,002$

Servicekomponente	Dimensionen	Kosten [USD]
AgentCore Gateway — Such-API	10.000 Interaktionen × 0,025$ pro 1.000	0,25$
AgentCore Gateway — API-Aufrufe	50.000 Aufrufe × 0,005 USD pro 1.000	0,25$
AWS Lambda-Funktion	Variabel je nach Nutzung (typische Workloads)	5,00$ — 20,00$
Geschätzte zusätzliche Kosten insgesamt:		5,50$ bis 20,50$ pro Monat

Die Preise sind freibleibend. Vollständige Informationen zu den AgentCore Gateway-Preisen finden Sie unter Amazon Bedrock Pricing (Abschnitt AgentCore Gateway). Informationen zu den Lambda-Preisen finden Sie unter AWS Lambda Pricing.

Sicherheit

Wenn Sie Systeme auf der AWS-Infrastruktur aufbauen, werden Sie und AWS gemeinsam für die Sicherheit verantwortlich sein. Dieses Modell der geteilten Verantwortung reduziert Ihren betrieblichen Aufwand, da AWS die Komponenten wie das Host-Betriebssystem, die Virtualisierungsebene und die physische Sicherheit der Einrichtungen, in denen die Services betrieben werden, betreibt, verwaltet und kontrolliert. Weitere Informationen zur AWS-Sicherheit finden Sie unter AWS Cloud Security.

IAM-Rollen

AWS Identity and Access Management (IAM) -Rollen ermöglichen es Kunden, Services und Benutzern in der AWS-Cloud detaillierte Zugriffsrichtlinien und -berechtigungen zuzuweisen. Diese Lösung erstellt IAM-Rollen, die den AWS-Lambda-Funktionen der Lösung Zugriff gewähren, um regionale Ressourcen zu erstellen.

Amazon CloudFront

Diese Lösung stellt eine Web-Benutzeroberfläche bereit, die in einem Amazon S3 S3-Bucket gehostet wird, der von Amazon CloudFront vertrieben wird. Um die Latenz zu reduzieren und

die Sicherheit zu verbessern, umfasst diese Lösung eine CloudFront Distribution mit einer Ursprungszugriffsidentität. Dabei handelt es sich um einen CloudFront Benutzer, der öffentlichen Zugriff auf die Bucket-Inhalte der Lösungswebsite gewährt. Standardmäßig verwendet die CloudFront Distribution TLS 1.2, um die höchste Sicherheitsstufe durchzusetzen. Weitere Informationen finden Sie unter Beschränken des Zugriffs auf einen Amazon S3 S3-Ursprung im Amazon CloudFront Developer Guide.

CloudFront aktiviert zusätzliche Sicherheitsmaßnahmen, um HTTP-Sicherheitsheader an jede Zuschauerantwort anzuhängen. Weitere Informationen finden Sie unter Hinzufügen oder Entfernen von HTTP-Headern in Antworten. CloudFront

Diese Lösung verwendet das CloudFront Standardzertifikat, für das mindestens das Sicherheitsprotokoll TLS v1.0 unterstützt wird. Um die Verwendung von TLS v1.2 oder TLS v1.3 zu erzwingen, müssen Sie ein benutzerdefiniertes SSL-Zertifikat anstelle des Standardzertifikats verwenden. CloudFront Weitere Informationen finden Sie unter Wie konfiguriere ich meine CloudFront Distribution für die Verwendung eines SSL/TLS Zertifikats?

Amazon API Gateway

Diese Lösung stellt Edge-optimierte Amazon API Gateway Gateway-Endpunkte bereit, um die Lasttestfunktion unter Verwendung des Standard-API-Gateway-Endpunkts anstelle einer benutzerdefinierten Domain bereitzustellen RESTful APIs . Für Edge-Optimierungen APIs mit dem Standardendpunkt verwendet API Gateway die TLS-1-0-Sicherheitsrichtlinie. Weitere Informationen finden Sie unter Working with REST APIs im Amazon API Gateway Developer Guide.

Diese Lösung verwendet das standardmäßige API-Gateway-Zertifikat, für das mindestens das Sicherheitsprotokoll TLS v1.0 unterstützt wird. Um die Verwendung von TLS v1.2 oder TLS v1.3 zu erzwingen, müssen Sie anstelle des standardmäßigen API-Gateway-Zertifikats eine benutzerdefinierte Domain mit einem benutzerdefinierten SSL-Zertifikat verwenden. Weitere Informationen finden Sie unter Benutzerdefinierte Domainnamen für REST einrichten. APIs

AWS Fargate-Sicherheitsgruppe

Standardmäßig öffnet diese Lösung die ausgehende Regel der AWS Fargate-Sicherheitsgruppe für die Öffentlichkeit. Wenn Sie verhindern möchten, dass AWS Fargate überall Datenverkehr sendet, ändern Sie die ausgehende Regel in ein bestimmtes Classless Inter-Domain Routing (CIDR).

Diese Sicherheitsgruppe umfasst auch eine Regel für eingehenden Datenverkehr, die lokalen Datenverkehr auf Port 50.000 zu jeder Quelle zulässt, die zu derselben Sicherheitsgruppe gehört. Dies wird verwendet, damit die Container miteinander kommunizieren können.

Amazon VPC

VPC: Eine virtuelle private Cloud (VPC), die auf dem Amazon VPC-Service basiert, bietet Ihnen ein privates, logisch isoliertes Netzwerk in der AWS-Cloud.

Sie können während der Bereitstellung Ihre eigene VPC in CloudFormation AWS-Parametern angeben. Die VPC wird ausschließlich von den ECS-Aufgaben verwendet, die Last erzeugen. Die Webkonsole und die API werden nicht in dieser VPC bereitgestellt. Wenn Sie keine vorhandene VPC angeben, erstellt die Lösung eine neue VPC mit der erforderlichen Netzwerkkonfiguration. Wenn Sie sich für die Verwendung einer vorhandenen VPC entscheiden, muss diese die folgenden Anforderungen erfüllen, um Lasttestaufgaben erfolgreich ausführen zu können.

VPC-Anforderungen

Die Mindestanforderungen für eine VPC zur Verwendung mit Distributed Load Testing auf AWS sind unten aufgeführt.

- Die VPC muss mindestens zwei enthalten AZs
- Die VPC muss mindestens zwei Subnetze enthalten, jedes in einer separaten AZ
- VPC-Subnetze können entweder öffentlich oder privat sein, sie müssen jedoch dieselbe Konfiguration verwenden (beide öffentlich ODER beide privat)
- Die VPC muss Zugriff auf Endpunkte für ECR, CloudWatch Logs, S3 und IoT Core bieten.
- Die VPC muss Zugriff auf die Dienste gewähren, auf die die Lasttests abzielen.

> **Note**
>
> Wenn Sie keine VPC haben, die diese Kriterien erfüllt, können Sie mit dem VPC-Assistenten schnell eine VPC erstellen. Weitere Informationen finden Sie unter Erstellen einer VPC.

Öffentliche Subnetze können diese Anforderungen erfüllen, indem sie Folgendes beinhalten:

- Ein an die VPC angeschlossenes Internet-Gateway
- Eine Route zum Internet-Gateway (0.0.0.0/0)

Private Subnetze können diese Anforderungen durch die Verwendung von NAT-Gateways oder VPC-Endpunkten erfüllen, wie unten beschrieben.

Option 1: NAT-Gateway

- Stellen Sie in jeder AZ mit privaten Subnetzen ein NAT-Gateway bereit
- Konfigurieren Sie Routing-Tabellen für die Weiterleitung von internetgebundenem Datenverkehr (0.0.0.0/0) über das NAT-Gateway

Option 2: VPC-Endpunkte

Erstellen Sie die folgenden VPC-Endpoints in Ihrer VPC:

- Amazon ECR API-Endpunkt: `com.amazonaws.<region>.ecr.api`
- Amazon ECR DKR-Endpunkt: `com.amazonaws.<region>.ecr.dkr`
- Amazon CloudWatch Logs-Endpunkt: `com.amazonaws.<region>.logs`
- Amazon S3 Gateway-Endpunkt: `com.amazonaws.<region>.s3`
- AWS IoT Core Core-Endpunkt (erforderlich, wenn Sie die Live-Datendiagramme verwenden) `com.amazonaws.<region>.iot.data`

Andere VPC-Konfigurationen funktionieren möglicherweise ebenfalls.

> **Important**
>
> Die Sicherheitsgruppe, die an jede VPC-Endpunktschnittstelle angehängt ist, muss eingehenden TCP-Verkehr auf Port 443 von der ECS-Aufgabensicherheitsgruppe zulassen.

Konfiguration der Sicherheitsgruppe

Während der Bereitstellung erstellt die Lösung eine Sicherheitsgruppe innerhalb Ihrer VPC, um den folgenden Datenverkehr mit Aufgaben im ECS-Cluster zuzulassen:

- Der gesamte ausgehende Verkehr
- Eingehender Datenverkehr auf Port 50000 von anderen Aufgaben in derselben Sicherheitsgruppe, um die Koordination zwischen den Aufgaben des Mitarbeiters und des Leiters zu erleichtern.

Netzwerk-Stresstest

Sie sind dafür verantwortlich, diese Lösung gemäß der Netzwerkstresstest-Richtlinie zu verwenden. Diese Richtlinie gilt beispielsweise für Situationen, in denen Sie planen, Netzwerktests mit hohem Volumen direkt von Ihren Amazon EC2 EC2-Instances an andere Standorte wie andere Amazon EC2 EC2-Instances, AWS-Eigenschaften/Services oder externe Endpunkte durchzuführen. Diese Tests werden manchmal als Stresstests, Belastungstests oder Gameday-Tests bezeichnet. Die meisten Kundentests fallen nicht unter diese Richtlinie. Beziehen Sie sich jedoch auf diese Richtlinie, wenn Sie glauben, dass Sie Traffic generieren, der insgesamt länger als 1 Minute über 1 Gbit/s (1 Milliarde Bit pro Sekunde) oder über 1 Gpps (1 Milliarde Pakete pro Sekunde) anhält.

Beschränkung des Zugriffs auf die öffentliche Benutzeroberfläche

Verwenden Sie die Sicherheitsautomatisierungslösung AWS WAF (Web Application Firewall), um den Zugriff auf die öffentlich zugängliche Benutzeroberfläche über die von IAM und Amazon Cognito bereitgestellten Authentifizierungs- und Autorisierungsmechanismen hinaus einzuschränken.

Diese Lösung stellt automatisch eine Reihe von AWS-WAF-Regeln bereit, die häufig vorkommende webbasierte Angriffe filtern. Benutzer können aus vorkonfigurierten Schutzfunktionen wählen, die die Regeln definieren, die in einer AWS WAF Web Access Control List (Web ACL) enthalten sind.

MCP-Serversicherheit (optional)

Wenn Sie die optionale MCP-Serverintegration einsetzen, verwendet die Lösung AWS AgentCore Gateway, um KI-Agenten einen sicheren Zugriff auf Lasttestdaten zu ermöglichen. AgentCore Gateway validiert Amazon Cognito Cognito-Authentifizierungstoken für jede Anfrage und stellt so sicher, dass nur autorisierte Benutzer auf den MCP-Server zugreifen können. Die Lambda-Funktion des MCP Servers implementiert Nur-Lese-Zugriffsmuster und verhindert so, dass KI-Agenten Testkonfigurationen oder -ergebnisse ändern. Für alle Interaktionen mit dem MCP-Server gelten dieselben Berechtigungsgrenzen und Zugriffskontrollen wie für die Webkonsole.

Unterstützte AWS Regionen

Diese Lösung verwendet den Amazon Cognito-Service, der derzeit nicht in allen AWS-Regionen verfügbar ist. Die aktuelle Verfügbarkeit von AWS-Services nach Regionen finden Sie in der regionalen AWS-Serviceliste.

Distributed Load Testing auf AWS ist in den folgenden AWS-Regionen verfügbar:

Name der Region	
USA Ost (Ohio)	Asien-Pazifik (Tokio)
USA Ost (Nord-Virginia)	Kanada (Zentral)
USA West (Nordkalifornien)	Europa (Frankfurt)
USA West (Oregon)	Europa (Irland)
Asien-Pazifik (Mumbai)	Europa (London)
Asien-Pazifik (Seoul)	Europa (Paris)
Asien-Pazifik (Singapur)	Europa (Stockholm)
Asien-Pazifik (Sydney)	Südamerika (São Paulo)

Von MCP Server unterstützte AWS-Regionen (optional)

Wenn Sie die optionale MCP-Serverintegration bereitstellen möchten, müssen Sie die Lösung in einer AWS-Region bereitstellen, in der AWS AgentCore Gateway verfügbar ist. Die MCP-Serverfunktion ist nur in den folgenden AWS-Regionen verfügbar:

Name der Region	Regionscode
USA Ost (Nord-Virginia)	us-east-1
USA West (Oregon)	us-west-2
Asien-Pazifik (Singapur)	ap-southeast-1
Asien-Pazifik (Sydney)	ap-southeast-2
Asien-Pazifik (Tokio)	ap-northeast-1
Europa (Frankfurt)	eu-central-1
Europa (Irland)	eu-west-1

Name der Region	Regionscode
Europa (London)	eu-west-2
Europa (Paris)	eu-west-3

Die aktuelle Verfügbarkeit von AWS AgentCore Gateway nach Regionen finden Sie unter AWS AgentCore Gateway-Endpunkte und Kontingente im AWS AgentCore Gateway Developer Guide.

Kontingente

Service Quotas, auch als Limits bezeichnet, sind die maximale Anzahl von Serviceressourcen oder -vorgängen für Ihr AWS-Konto.

Kontingente für AWS-Services in dieser Lösung

Stellen Sie sicher, dass Sie über ein ausreichendes Kontingent für jeden der in dieser Lösung implementierten Services verfügen. Weitere Informationen finden Sie unter AWS-Servicekontingente.

Verwenden Sie die folgenden Links, um zur Seite für diesen Dienst zu gelangen. Um die Service-Kontingente für alle AWS-Services in der Dokumentation anzuzeigen, ohne zwischen den Seiten zu wechseln, sehen Sie sich stattdessen die Informationen auf der Seite Service-Endpunkte und Kontingente in der PDF-Datei an.

CloudFormation AWS-Kontingente

Ihr AWS-Konto verfügt über CloudFormation AWS-Kontingente, die Sie beachten sollten, wenn Sie den Stack in dieser Lösung starten. Wenn Sie diese Kontingente verstehen, können Sie Limitationsfehler vermeiden, die Sie daran hindern würden, diese Lösung erfolgreich einzusetzen. Weitere Informationen finden Sie unter CloudFormation AWS-Kontingente im CloudFormation AWS-Benutzerhandbuch.

Kontingente für Lasttests

Die maximale Anzahl von Aufgaben, die in Amazon ECS mit dem Starttyp AWS Fargate ausgeführt werden können, basiert auf der vCPU-Größe der Aufgaben. Die Standardaufgabengröße bei Distributed Load Testing auf AWS beträgt 2 vCPU. Die aktuellen Standardkontingente finden Sie

unter Amazon ECS-Servicekontingente. Die Kontingente für aktuelle Konten können von den aufgeführten Kontingenten abweichen. Um die für ein Konto spezifischen Kontingente zu überprüfen, überprüfen Sie das Service-Kontingent für die Anzahl der On-Demand-vCPU-Ressourcen von Fargate in der AWS-Managementkonsole. Anweisungen, wie Sie eine Erhöhung beantragen können, finden Sie unter AWS-Servicekontingente im AWS General Reference Guide.

Das Amazon Linux 2023 Container-Image (mit installiertem Taurus) schränkt die gleichzeitigen Verbindungen pro Aufgabe nicht ein. Dies bedeutet jedoch nicht, dass es eine unbegrenzte Anzahl von Benutzern unterstützen kann. Informationen zur Bestimmung der Anzahl gleichzeitiger Benutzer, die die Container für einen Test generieren können, finden Sie im Abschnitt „Anzahl der Benutzer ermitteln“ dieses Handbuchs.

> **Note**
>
> Das empfohlene Limit für gleichzeitige Benutzer liegt auf der Grundlage der Standardeinstellungen bei 200 Benutzern.

Gleichzeitige Tests

Diese Lösung erstellt für jeden Test ein CloudWatch Amazon-Dashboard, das die kombinierte Ausgabe aller im Amazon ECS-Cluster ausgeführten Aufgaben in Echtzeit anzeigt. Das CloudWatch Dashboard zeigt die durchschnittliche Antwortzeit, die Anzahl der gleichzeitigen Benutzer, die Anzahl der erfolgreichen Anfragen und die Anzahl der fehlgeschlagenen Anfragen. Die Lösung aggregiert jede Metrik sekundengenau und aktualisiert das Dashboard jede Minute.

Amazon EC2-Testrichtlinie

Sie benötigen keine Genehmigung von AWS, um Lasttests mit dieser Lösung durchzuführen, solange Ihr Netzwerkverkehr unter 1 Gbit/s bleibt. Wenn Ihr Test mehr als 1 Gbit/s generiert, wenden Sie sich an AWS. Weitere Informationen finden Sie in der Amazon EC2-Testrichtlinie.

Amazon-Richtlinie für CloudFront Auslastungstests

Wenn Sie planen, einen CloudFront Endpunkt unter Last zu testen, lesen Sie die Richtlinien für Auslastungstests im Amazon CloudFront Developer Guide. Wir empfehlen außerdem, den Traffic auf mehrere Aufgaben und Regionen zu verteilen. Stellen Sie mindestens 30 Minuten Anlaufzeit für den Auslastungstest bereit. Für Auslastungstests, bei denen mehr als 500.000 Anfragen pro

Sekunde gesendet werden oder mehr als 300 Gbit/s Daten benötigt werden, empfehlen wir, zunächst eine Vorabgenehmigung für das Senden des Datenverkehrs einzuholen. CloudFront kann nicht genehmigten Lasttestverkehr drosseln, der sich auf die Verfügbarkeit der Dienste auswirkt. CloudFront

Überwachung der Lösung nach der Bereitstellung

Nach der Bereitstellung der Lösung empfehlen wir, die Ressourcen der Lösung mithilfe von CloudWatch Amazon-Alarmen und -Metriken kontinuierlich zu überwachen.

CloudWatch Alarme einrichten

Sie können CloudWatch Alarme einrichten, um wichtige Kennzahlen zu überwachen und Benachrichtigungen zu erhalten, wenn Schwellenwerte überschritten werden. Erwägen Sie die Einrichtung von Alarmen für die folgenden Ressourcen:

CloudFront Amazon-Vertriebsmetriken

Überwachen Sie die CloudFront Vertriebsleistung und die Fehler. Weitere Informationen finden Sie unter CloudFront Vertriebsmetriken im Amazon CloudFront Developer Guide.

Metriken für Amazon API Gateway

Überwachen Sie API-Anforderungsraten, Latenz und Fehler. Weitere Informationen finden Sie unter Amazon API Gateway-Dimensionen und -Metriken im Amazon API Gateway Developer Guide.

AWS Lambda Lambda-Funktionsmetriken

Überwachen Sie Lambda-Funktionsaufrufe, Dauer, Fehler und Drosselungen für die Microservices der Lösung.

Amazon ECS- und AWS Fargate-Metriken

Überwachen Sie die CPU- und Speicherauslastung von Aufgaben während der Lasttests, um sicherzustellen, dass die Ressourcen ausreichend sind.

Amazon DynamoDB-Metriken

Überwachen Sie den Lese- und Schreibkapazitätsverbrauch, gedrosselte Anfragen und die Latenz.

Beauftragen Sie einen Experten

Kurzfristige AWS Countdown Premium-Engagements für verteilte Lasttests auf AWS

Unsere AWS-Techniker bieten fachkundige Beratung zu den Grundlagen von Leistungstests, zur Skriptentwicklung und zur Ergebnisanalyse. Melden Sie sich jetzt an.

Übersicht

AWS Countdown Premium (CDP) Short Term Engagements bieten Expertenberatung für Unternehmen, die Leistungstests in großem Umfang durchführen. Durch ein kollaboratives do-it-yourself „" Modell bieten AWS-Techniker strategische Aufsicht und technisches Fachwissen, während Ihr Team die Verantwortung für die Ausführung behält. Erfahrene AWS-Techniker stehen innerhalb einer Woche nach der Registrierung zur Verfügung, ohne dass langfristige Verträge erforderlich sind.

Servicemodell

CDP-Techniker arbeiten mit Ihrem Team zusammen, um Sie während der gesamten Implementierung Ihrer Leistungstests zu beraten und zu beaufsichtigen. Dieser praxisorientierte Ansatz stellt sicher, dass Sie fachkundige Anleitung erhalten und gleichzeitig interne Fähigkeiten aufbauen. Der Service ist ideal für Unternehmen mit bestehenden Testkapazitäten, die spezielles AWS-Fachwissen benötigen, um Distributed Load Testing auf AWS effektiv zu implementieren.

Was bieten CDP-Techniker

CDP-Techniker führen Sie durch die Grundlagen von Leistungstests und Distributed Load Testing auf der AWS-Architektur. Sie bieten Anleitungen zur JMeter K6- und Locust-Skriptstruktur und zur Entwicklung von Testskripten, unterstützen bei der Bereitstellung von CloudFormation Vorlagen und bewerten die Testergebnisse mit Empfehlungen zur Leistungsoptimierung. Der Support umfasst die Analyse der Ressourcennutzung, die Abstimmung von Best Practices und die end-to-end Beratung von der ersten Einrichtung bis hin zur Ergebnisanalyse, sodass der Wissenstransfer an Ihr Team ermöglicht wird.

Pflichten des Kunden

Ihr Team kümmert sich um Konfigurationen auf Anwendungsebene, die Entwicklung von Testskripten und die Überprüfung von Testszenarien. Sie behalten die Verantwortung für die tatsächliche Ausführung und den Betrieb der Tests, einschließlich aller Testaktivitäten vor, während und nach Leistungstests.

Wichtigste Vorteile

Kurzfristige CDP-Engagements reduzieren das Risiko durch fachkundige Aufsicht, kontextbezogene Beratung speziell für Ihre Arbeitslast, Empfehlungen zur Leistungsoptimierung, schnellere Problemlösung, Abstimmung bewährter Verfahren und umfassenden Support, während gleichzeitig die Eigenverantwortung und die Weiterentwicklung der Fähigkeiten Ihres Teams gewahrt bleiben.

Unterstützte Architekturen

Distributed Load Testing auf AWS unterstützt Tests für Webanwendungen APIs, Microservices und serverlose Architekturen in großem Maßstab und nutzt dabei die Distributed Load Testing on AWS-Lösung. Die Testmöglichkeiten gehen weit über diese üblichen Anwendungsfälle hinaus und umfassen Datenbanken, TCP/UDP Protokolle, LDAP-Verzeichnisse, SMTP-Mailserver und viele andere Systeme und Protokolle, die eine Leistungsvalidierung unter Last erfordern.

Erste Schritte

Organizations, die an CDP Short Term Engagements for Distributed Load Testing auf AWS interessiert sind, können sich hier direkt über die AWS-Website anmelden und „Use Case Implementation“ für Ihren Schwerpunktbereich auswählen.

Außerhalb des Geltungsbereichs

CDP bietet keine Entwicklung von benutzerdefinierten Testskripten (nur Anleitungen), verwaltet keine Testausführungen und organisiert auch keine kundenspezifischen praktischen Übungen oder Workshops. Support vor Ort fällt ebenfalls nicht in den Geltungsbereich.

Bereitstellen der Lösung

AWS Launch Wizard ist die empfohlene Bereitstellungsmethode für diese Lösung. Es bietet:

- Ein geführtes Konfigurationserlebnis mit detaillierten Hilfefenstern bei jedem Schritt
- Eine zentrale Seite zur Überwachung des Zustands all Ihrer Bereitstellungen
- Angabe, wann eine neuere Version der Lösung für die Bereitstellung oder Aktualisierung verfügbar ist

Alternativ können Sie die Lösung direkt mithilfe einer CloudFormation AWS-Vorlage bereitstellen.

Überblick über den Bereitstellungsprozess

Bevor Sie die Lösung bereitstellen, sollten Sie sich mit den Kosten, der Architektur, der Sicherheit und anderen Überlegungen befassen, die weiter oben in diesem Handbuch erörtert wurden.

Bereitstellungszeit: Ungefähr 15 Minuten für den Haupt-Stack plus 5 Minuten für jede weitere Region

> **Note**
>
> Diese Lösung beinhaltet Datenerfassungsmetriken für AWS. Wir verwenden diese Daten, um besser zu verstehen, wie Kunden diese Lösung und die damit verbundenen Services und Produkte nutzen. AWS ist Eigentümer der im Rahmen dieser Umfrage gesammelten Daten. Die Datenerfassung unterliegt der AWS-Datenschutzerklärung.

> **Note**
>
> Sie sind für die Kosten der AWS-Services verantwortlich, die beim Betrieb dieser Lösung verwendet werden. Weitere Informationen finden Sie im Abschnitt Kosten in diesem Handbuch und auf der Preisseite für jeden AWS-Service, der in dieser Lösung verwendet wird.

Bereitstellung mit dem AWS Launch Wizard

Diese Lösung umfasst einen geführten Bereitstellungsprozess mithilfe des AWS Launch Wizard. Gehen Sie wie folgt vor, um Distributed Load Testing auf AWS in Ihrem Konto bereitzustellen.

1. Melden Sie sich bei der AWS-Managementkonsole an und klicken Sie auf die Schaltfläche unten, um den Bereitstellungsprozess zu starten.

2. Wenn für die Lösung mehr als ein Bereitstellungsmuster verfügbar ist, wählen Sie das aus, das für Ihren Anwendungsfall am besten geeignet ist.
3. Wählen Sie eine Version für die Bereitstellung aus. Die neueste Version wird empfohlen.
4. Klicken Sie auf die Schaltfläche Deployment Wizard starten.

Anschließend folgen Sie einer Reihe von Schritten, um die Informationen zu sammeln, die für die Bereitstellung der Lösung erforderlich sind. Die Bereitstellung der erforderlichen Ressourcen dauert ungefähr 15 Minuten.

Wählen Sie Ihre Bereitstellung aus der Bereitstellungsliste aus, um deren Status einzusehen.

Mit AWS bereitstellen CloudFormation

Diese Lösung verwendet CloudFormation AWS-Vorlagen und Stacks, um ihre Bereitstellung zu automatisieren. Die CloudFormation Vorlagen spezifizieren die in dieser Lösung enthaltenen AWS-Ressourcen und ihre Eigenschaften. Der CloudFormation Stack stellt die Ressourcen bereit, die in den Vorlagen beschrieben sind.

CloudFormation AWS-Vorlage

Sie können die CloudFormation Vorlage für diese Lösung herunterladen, bevor Sie sie bereitstellen. Diese Lösung verwendet AWS CloudFormation , um die Bereitstellung von Distributed Load Testing auf AWS zu automatisieren. Es enthält die folgende CloudFormation AWS-Vorlage, die Sie vor der Bereitstellung herunterladen können:

View template

distribu

load-testing-on-aws.template — Verwenden Sie diese Vorlage, um die Lösung und alle zugehörigen Komponenten zu starten. In der Standardkonfiguration werden die Kern- und Unterstützungsdienste bereitgestellt, die in den AWS-Services in diesem Lösungsabschnitt enthalten sind. Sie können die Vorlage jedoch an Ihre spezifischen Anforderungen anpassen.

> **Note**
>
> CloudFormation AWS-Ressourcen werden aus Konstrukten des AWS Cloud Development Kit (AWS CDK) erstellt. Wenn Sie diese Lösung bereits bereitgestellt haben, finden Sie Anweisungen zum Update unter Lösung aktualisieren.

Starten des -Stacks

Gehen Sie wie folgt vor, um die Lösung Distributed Load Testing on AWS in Ihrem Konto bereitzustellen. Diese automatisierte CloudFormation AWS-Vorlage stellt Distributed Load Testing auf AWS bereit.

1. Melden Sie sich bei der AWS-Managementkonsole an und klicken Sie auf die Schaltfläche, um die CloudFormation Vorlage zu starten.

   Launch solution

   Alternativ können Sie die Vorlage als Ausgangspunkt für Ihre eigene Implementierung herunterladen.

2. Die Vorlage wird standardmäßig in der Region USA Ost (Nord-Virginia) gestartet. Um diese Lösung in einer anderen AWS-Region zu starten, verwenden Sie die Regionsauswahl in der Navigationsleiste der Konsole.

   > **Note**
   >
   > Diese Lösung verwendet Amazon Cognito, das derzeit nur in bestimmten AWS-Regionen verfügbar ist. Daher müssen Sie diese Lösung in einer AWS-Region starten, in der

Amazon Cognito verfügbar ist. Die aktuelle Serviceverfügbarkeit nach Regionen finden Sie in der regionalen AWS-Serviceliste.

3. Vergewissern Sie sich auf der Seite Stack erstellen, dass die richtige Vorlagen-URL im Textfeld Amazon S3 S3-URL angezeigt wird, und wählen Sie Weiter.
4. Weisen Sie Ihrem Lösungsstapel auf der Seite „Stack-Details angeben“ einen Namen zu.
5. Überprüfen Sie unter Parameter die Parameter für die Vorlage und ändern Sie sie nach Bedarf. Diese Lösung verwendet die folgenden Standardwerte.

Parameter	Standard	Description
Name des Administrators	<Erfordert Eingabe>	Benutzername für den ursprünglichen Lösungsad ministrator.
E-Mail-Adresse des Administr ators	*<Requires input>*	E-Mail-Adresse des Administr atorbenutzers. Nach dem Start wird eine E-Mail mit Anweisungen zur Anmeldung auf der Konsole an diese Adresse gesendet.
Bestehende VPC-ID	<Optional input>	Wenn Sie über eine VPC verfügen, die Sie verwenden möchten und die bereits erstellt wurde, geben Sie die ID einer vorhandenen VPC in derselben Region ein, in der der Stack bereitges tellt wurde. Zum Beispiel vpc-1a2b3c4d5e6f.
Erstes vorhandenes Subnetz	<Optional input>	Die ID des ersten Subnetzes innerhalb Ihrer vorhanden en VPC. Dieses Subnetz benötigt eine Route zum Internet, um das Container

Parameter	Standard	Description
		-Image für die Ausführung von Tests abzurufen. Zum Beispiel subnet-7h8i9j0k.
Zweites vorhandenes Subnetz	<Optional input>	Die ID des zweiten Subnetzes innerhalb der vorhandenen VPC. Dieses Subnetz benötigt eine Route zum Internet, um das Container-Image für die Ausführung von Tests abzurufen. Zum Beispiel subnet-1x2y3z.
Geben Sie einen gültigen CIDR-Block für die Lösung an, um VPC zu erstellen	192.168.0.0/16	Sie können diesen Parameter leer lassen, wenn Sie eine bestehende VPC verwenden.
Geben Sie einen gültigen CIDR-Block für Subnetz A an, damit die Lösung eine VPC erstellen kann	192.168.0.0/20	CIDR-Block für Subnetz A der AWS Fargate VPC
Geben Sie einen gültigen CIDR-Block für Subnetz B an, damit die Lösung eine VPC erstellen kann	192.168.16.0/20	CIDR-Block für Subnetz B der AWS Fargate VPC
Stellen Sie einen CIDR-Block bereit, um den ausgehenden Verkehr von Fargate-Aufgaben zuzulassen	0.0.0.0/0	CIDR-Block, der den ausgehenden Zugriff auf Amazon ECS-Container einschränkt.

Parameter	Standard	Description
Container-Image automatisch aktualisieren	No	Verwenden Sie bis zur nächsten Nebenversion automatisch das aktuellste und sicherste Image. Bei Auswahl dieser Option No wird das Image in der ursprünglich veröffentlichten Version ohne Sicherheitsupdates abgerufen.
Stellen Sie den optionalen MCP-Server bereit	No	Stellen Sie den optionalen Remote-MCP-Server bereit und verwenden Sie AgentCore Gateway, um KI-Anwendungen mit Distributed Load Testing auf AWS zu verbinden.

6. Wählen Sie Weiter aus.
7. Wählen Sie auf der Seite Configure stack options (Stack-Optionen konfigurieren) Next (Weiter) aus.
8. Überprüfen und bestätigen Sie die Einstellungen auf der Seite Review. Markieren Sie das Kästchen, um zu bestätigen, dass die Vorlage AWS Identity and Access Management (IAM) -Ressourcen erstellt.
9. Wählen Sie Stack erstellen aus, um den Stack bereitzustellen.

Sie können den Status des Stacks in der CloudFormation AWS-Konsole in der Spalte Status anzeigen. Sie sollten in etwa 15 Minuten den Status CREATE_COMPLETE erhalten.

> **Note**
>
> Zusätzlich zur primären AWS-Lambda-Funktion umfasst diese Lösung die Lambda-Funktion für benutzerdefinierte Ressourcen, die nur während der Erstkonfiguration oder wenn Ressourcen aktualisiert oder gelöscht werden, ausgeführt wird.

Beim Ausführen dieser Lösung ist die Lambda-Funktion für benutzerdefinierte Ressourcen inaktiv. Löschen Sie diese Funktion jedoch nicht, da sie für die Verwaltung der zugehörigen Ressourcen erforderlich ist.

Bereitstellung in mehreren Regionen

Bereitstellungszeit: Ungefähr 5 Minuten pro Region

Sie können Tests in mehreren Regionen durchführen.

Wenn Sie die Distributed Load Testing-Lösung bereitstellen, erstellt sie eine regionale CloudFormation Vorlage im S3-Bucket für Szenarien. Die URL für diese Vorlage ist in den CloudFormation Ausgaben Ihres Haupt-Stacks unter dem Schlüssel „RegionalCFTemplate“ aufgeführt.

Um einen Test mit mehreren Regionen durchzuführen, müssen Sie die regionale CloudFormation Vorlage in jeder Region bereitstellen, in der Sie den Test ausführen möchten.

> **Note**
>
> Jedes AWS-Konto kann nur einen regionalen Stack pro Region verwenden. Außerdem kann der regionale Stack nicht in derselben Region wie der Haupt-Stack verwendet werden.

Sie können die regionale Vorlage wie folgt installieren:

1. Navigieren Sie in der Webkonsole der Lösung im linken Menü zu Dashboard.
2. Verwenden Sie das Zwischenablage-Symbol, um den CloudFormation Vorlagenlink in Amazon S3 zu kopieren.
3. Melden Sie sich bei der CloudFormation AWS-Konsole an und wählen Sie die richtige Region aus.
4. Vergewissern Sie sich auf der Seite Stack erstellen, dass die richtige Vorlagen-URL im Textfeld Amazon S3 S3-URL angezeigt wird, und wählen Sie Weiter.
5. Weisen Sie Ihrem Lösungsstapel auf der Seite „Stack-Details angeben“ einen Namen zu.
6. Überprüfen Sie unter Parameter die Parameter für die Vorlage und ändern Sie sie nach Bedarf. Diese Lösung verwendet die folgenden Standardwerte.

Parameter	Standard	Description
Bestehende VPC-ID	<Optional input>	Wenn Sie über eine VPC verfügen, die Sie verwenden möchten und die bereits erstellt wurde, geben Sie die ID einer vorhandenen VPC in derselben Region ein, in der der Stack bereitgestellt wurde. Zum Beispiel vpc-1a2b3c4d5e6f.
Erstes vorhandenes Subnetz	<Optional input>	Die ID des ersten Subnetzes innerhalb Ihrer vorhandenen VPC. Dieses Subnetz benötigt eine Route zum Internet, um das Container-Image für die Ausführung von Tests abzurufen. Zum Beispiel subnet-7h8i9j0k.
Zweites vorhandenes Subnetz	<Optional input>	Die ID des zweiten Subnetzes innerhalb der vorhandenen VPC. Dieses Subnetz benötigt eine Route zum Internet, um das Container-Image für die Ausführung von Tests abzurufen. Zum Beispiel subnet-1x2y3z.

Parameter	Standard	Description
Geben Sie einen gültigen CIDR-Block für die Lösung an, um VPC zu erstellen	192.168.0.0/16	Wenn Sie keine Werte für eine bestehende VPC angeben, enthält der CIDR-Block für die von der Lösung erstellte Amazon-VPC die IP-Adresse für AWS Fargate.
Stellen Sie einen CIDR-Block bereit, um den ausgehenden Verkehr von Fargate-Aufgaben zuzulassen	0.0.0.0/0	CIDR-Block, der den ausgehenden Zugriff auf Amazon ECS-Container einschränkt.

7. Wählen Sie Weiter aus.
8. Wählen Sie auf der Seite Configure stack options (Stack-Optionen konfigurieren) Next (Weiter) aus.
9. Überprüfen und bestätigen Sie die Einstellungen auf der Seite Review. Stellen Sie sicher, dass Sie das Kästchen ankreuzen, um zu bestätigen, dass die Vorlage AWS Identity and Access Management (IAM) -Ressourcen erstellt.
10. Wählen Sie Stack erstellen aus, um den Stack bereitzustellen.

    Sie können den Status des Stacks in der CloudFormation AWS-Konsole in der Spalte Status anzeigen. Sie sollten in etwa fünf Minuten den Status CREATE_COMPLETE erhalten.

Wenn die Regionen erfolgreich bereitgestellt wurden, werden sie in der Webkonsole angezeigt. Wenn Sie einen Test erstellen, werden alle verfügbaren Regionen im Dashboard und unter Szenarioerstellung aufgelistet. Sie können einem Test im Schritt Traffic Shape der Szenarioerstellung eine Region hinzufügen.

Die Lösung erstellt ein DynamoDB-Element für jede bereitgestellte Region in der Szenariotabelle, das die erforderlichen Informationen zu den Testressourcen in dieser Region enthält. Sie können die Testergebnisse in der Webkonsole nach Region sortieren. Verwenden Sie CloudWatch Amazon-Metriken, um aggregierte Ergebnisse für alle Regionen in einem Test mit mehreren Regionen anzuzeigen. Den Quellcode für das Diagramm finden Sie nach Abschluss des Tests in den Testergebnissen.

> **Note**
>
> Sie können den regionalen Stack ohne die Webkonsole starten. Besorgen Sie sich einen Link zur regionalen Vorlage im Amazon S3 S3-Szenario-Bucket und geben Sie ihn als Quelle an, wenn Sie den regionalen Stack in der erforderlichen Region starten. Alternativ können Sie die Vorlage herunterladen und als Quelle für die gewünschte Region hochladen.

Aktualisieren Sie die Lösung

Bei der Aktualisierung der Lösung werden die neuesten Funktionen, Sicherheitspatches und Bugfixes auf Ihre Bereitstellung angewendet. Informationen zum Update auf die neueste Version finden Sie im entsprechenden Abschnitt, der auf Ihrer ursprünglichen Bereitstellungsmethode basiert: AWS Launch Wizard oder AWS CloudFormation.

> **Important**
>
> Stellen Sie vor dem Update sicher, dass derzeit keine Auslastungstests ausgeführt werden. Der Aktualisierungsprozess kann die Verfügbarkeit der Lösung vorübergehend beeinträchtigen.

Aktualisierung mit dem AWS Launch Wizard

Die Konsole zeigt automatisch die neueste verfügbare Version der Lösung in der Dropdownliste Bereitstellungsversion an. Wenn Sie die Lösung bereits bereitgestellt haben, gehen Sie wie folgt vor, um Ihre Bereitstellung auf die neueste Version zu aktualisieren.

1. Gehen Sie zu Launch Wizard Deployments.
2. Wählen Sie die Bereitstellung aus, die Sie aktualisieren möchten.
3. Wählen Sie Aktionen und dann Bereitstellungsversion aktualisieren aus.
4. Wählen Sie die neueste Version aus den verfügbaren Bereitstellungsversionen aus.
5. Überprüfen Sie die Konfiguration.
6. Nehmen Sie bei jedem Schritt die erforderlichen Änderungen vor.
7. Bestätigen Sie das Update.

Update mit AWS CloudFormation

Wenn Sie die Lösung bereits bereitgestellt haben, gehen Sie wie folgt vor, um den CloudFormation Stack auf die neueste Version zu aktualisieren.

1. Melden Sie sich bei der CloudFormation Konsole an, wählen Sie Ihren vorhandenen CloudFormation Stack aus und wählen Sie Stack aktualisieren aus.

2. Wählen Sie Direktes Update durchführen aus.
3. Wählen Sie Bestehende Vorlage ersetzen aus.
4. Gehen Sie unter Vorlage angeben wie folgt vor:
   a. Wählen Sie Amazon S3 S3-URL aus.
   b. Kopieren Sie den Link der neuesten Vorlage.
   c. Fügen Sie den Link in das Amazon S3 S3-URL-Feld ein.
   d. Vergewissern Sie sich, dass die richtige Vorlagen-URL im Textfeld Amazon S3 S3-URL angezeigt wird.
   e. Wählen Sie Weiter aus.
   f. Wählen Sie erneut Next (Weiter).
5. Überprüfen Sie unter Parameter die Parameter für die Vorlage und ändern Sie sie nach Bedarf. Einzelheiten zu den Parametern finden Sie unter Den Stack starten.
6. Wählen Sie Weiter aus.
7. Wählen Sie auf der Seite Configure stack options (Stack-Optionen konfigurieren) Next (Weiter) aus.
8. Überprüfen und bestätigen Sie die Einstellungen auf der Seite Review.
9. Markieren Sie das Kästchen, das bestätigt, dass die Vorlage möglicherweise IAM-Ressourcen erstellt.
10. Wählen Sie „Änderungssatz anzeigen“ und überprüfen Sie die Änderungen.
11. Wählen Sie Stack aktualisieren, um den Stack bereitzustellen.

Sie können den Status des Stacks in der CloudFormation AWS-Konsole in der Spalte Status anzeigen. Sie sollten in etwa 15 Minuten einen `UPDATE_COMPLETE` Status erhalten.

> **Note**
>
> Wenn bei der Anmeldung über Ihren Browser nach dem Stack-Upgrade Probleme mit der Amazon Cognito Cognito-Authentifizierung auftreten, aktualisieren Sie bitte Ihren Browser (Strg+Shift+R auf Windows/Linux oder Cmd+Shift+R auf Mac), um die zwischengespeicherten Daten zu löschen, und versuchen Sie es erneut.

Fehlerbehebung bei Updates von Versionen vor v3.3.0

> **Note**
>
> Dieser Abschnitt gilt nur für Updates von Versionen vor v3.3.0. Wenn Sie von Version 3.3.0 oder höher aktualisieren, folgen Sie dem Standard-Aktualisierungsverfahren über den AWS Launch Wizard oder AWS. CloudFormation

1. Laden Sie die -aws.template distributed-load-testing-on herunter.
2. Öffnen Sie die Vorlage und navigieren Sie zu `Conditions:` und suchen Sie nach. `DLTCommonResourcesAppRegistryCondition`
3. Die Ausgabe sollte wie folgt aussehen:

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "true"
```

4. Ändern Sie den zweiten `true` Wert in`false`:

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "false"
```

5. Verwenden Sie die benutzerdefinierte Vorlage, um Ihren Stack zu aktualisieren, indem Sie die Schritte unter Update mit AWS ausführen CloudFormation.
6. Dieses Update entfernt Ressourcen, die sich auf die App-Registrierung beziehen, aus dem Stack, sodass das Update erfolgreich abgeschlossen werden kann.
7. Führen Sie ein weiteres Stack-Update mit der neuesten Vorlagen-URL durch.

Regionale Stacks werden aktualisiert

Wenn Sie die Lösung in mehreren Regionen eingesetzt haben, müssen Sie jeden regionalen Stack separat aktualisieren. Folgen Sie dem Standardaktualisierungsverfahren für jeden regionalen CloudFormation Stack in den Regionen, in denen Sie die Testinfrastruktur bereitgestellt haben.

AWS Systems Manager Manager-Anwendungsmanager

Nach der Aktualisierung der Lösung bietet AWS Systems Manager Application Manager einen Überblick über die Lösung und ihre Ressourcen auf Anwendungsebene. Sie können Application Manager verwenden, um:

- Überwachen Sie Ressourcen, Kosten für bereitgestellte Ressourcen in Stacks und AWS-Konten sowie Protokolle von einem zentralen Standort aus.
- Zeigen Sie Betriebsdaten für die Ressourcen der Lösung im Kontext einer Anwendung an, z. B. den Bereitstellungsstatus, CloudWatch Alarme, Ressourcenkonfigurationen und Betriebsprobleme.

Fehlerbehebung

Die Lösung bekannter Probleme enthält Anweisungen zur Behebung bekannter Fehler. Wenn diese Anweisungen Ihr Problem nicht lösen, finden Sie unter Wenden Sie sich an den AWS-Support. Dort finden Sie Anweisungen zum Öffnen einer AWS-Supportanfrage für diese Lösung.

Lösung eines bekannten Problems

Problem: Sie verwenden eine bestehende VPC und Ihre Tests schlagen mit dem Status Fehlgeschlagen fehl, was zu der folgenden Fehlermeldung führt:

```
Test might have failed to run.
```

- Auflösung

Stellen Sie sicher, dass die Subnetze in der angegebenen VPC vorhanden sind und dass sie über eine Route zum Internet entweder mit einem Internet-Gateway oder einem NAT-Gateway verfügen. AWS Fargate benötigt Zugriff, um das Container-Image aus dem öffentlichen Repository abzurufen, um Tests erfolgreich ausführen zu können.

Problem: Die Ausführung von Tests dauert zu lange oder läuft auf unbestimmte Zeit

- Auflösung

Brechen Sie den Test ab und überprüfen Sie AWS Fargate, um sicherzustellen, dass alle Aufgaben beendet wurden. Wenn sie nicht gestoppt wurden, beenden Sie alle Fargate-Aufgaben manuell. Überprüfen Sie die Fargate-Aufgabenlimits auf Abruf in Ihrem Konto, um sicherzustellen, dass Sie die gewünschte Anzahl von Aufgaben starten können. Sie können auch die CloudWatch Protokolle der Lambda-Task-Runner-Funktion überprüfen, um mehr über Fehler beim Starten von Fargate-Aufgaben zu erfahren. In den CloudWatch ECS-Protokollen finden Sie Einzelheiten darüber, was in laufenden Fargate-Containern passiert.

Problem: Die Tests werden gestartet, können aber nicht abgeschlossen werden oder der Status der ECS-Aufgaben ist unbekannt

- Auflösung

Wenn Sie die Option zur Bereitstellung einer vorhandenen VPC in dem Konto ausgewählt haben, in dem die Lösung bereitgestellt wurde, stellen Sie sicher, dass die von den ECS-Aufgaben verwendete VPC über genügend freie IP-Adressen verfügt, um die in der Testeingabe angegebene Anzahl von Aufgaben zu starten. Die ECS-Aufgabendefinition verwendet das ECR-Image, das ein Internet-Gateway oder eine Route zum Internet benötigt, sodass der ECS-Dienst die Aufgaben bereitstellen kann, indem er das Lösungs-ECR-Image von aws-solutions/ - herunterlädt. distributed-load-testing-on aws-load-tester Wenn Sie keine Route zum Internet bereitstellen können, da alle Subnetze in der VPC privat sind, können Sie das ECR-Image mithilfe des ECR-Pull-Through-Cache in Ihrem Konto hosten. Aktualisieren Sie die Aufgabendefinition mit der neuen ECR-Image-URI und erstellen Sie eine neue Revision. Sobald die Aufgabendefinition aktualisiert wurde, muss die Lösungskonfiguration in der DynamoDB-Tabelle aktualisiert werden, um die neue Version verwenden zu können. Den Namen der DynamoDB-Tabelle finden Sie auf der Registerkarte CloudFormation Stack-Ausgaben unter dem Schlüssel. ScenariosTable Aktualisieren Sie das Attribut taskDefinition für das Element mit dem Schlüssel TestID und dem Wert region- [SOLUTION-DEPLOYED-REGION].

Problem: Tests müssen einen Endpunkt verwenden, der privat ist oder nicht über das Internet-Gateway verfügbar ist

- Auflösung

Wenn Sie private API-Endpunkte testen, auf die nicht über das Internet-Gateway zugegriffen werden kann, sollten Sie die folgenden Ansätze in Betracht ziehen:

1. Netzwerkkonfiguration: Stellen Sie sicher, dass die von den ECS-Aufgaben verwendeten Subnetz-Routentabellen mit einer Route zum IP-Adressbereich des privaten Endpunkts aktualisiert werden, der getestet wird. Dadurch kann der Testdatenverkehr den privaten Endpunkt innerhalb Ihrer VPC erreichen.
2. DNS-Auflösung: Konfigurieren Sie für benutzerdefinierte Domänen die DNS-Einstellungen in Ihrer VPC, um den Domainnamen des privaten Endpunkts aufzulösen. Detaillierte Anweisungen finden Sie in der Dokumentation zu VPC DNS.
3. VPC-Endpunkte: Wenn Sie AWS-Services testen, sollten Sie die Verwendung von VPC-Endpunkten (AWS PrivateLink) in Betracht ziehen, um private Konnektivität herzustellen. Um beispielsweise ein privates API Gateway zu testen, können Sie einen VPC-Endpunkt für API Gateway erstellen. Weitere Informationen finden Sie in der Dokumentation zu Private API Gateway.
4. VPC-Peering: Wenn sich der private Endpunkt in einer anderen VPC befindet, richten Sie ein VPC-Peering zwischen der VPC, in der die Lösung bereitgestellt wird, und der VPC mit dem privaten

Endpunkt ein. Konfigurieren Sie in beiden Fällen die entsprechenden Routing-Tabellen. VPCs Weitere Informationen finden Sie in der VPC-Peering-Dokumentation.

5. Transit Gateway: Für komplexere Netzwerkszenarien, die mehrere umfassen VPCs, sollten Sie die Verwendung von AWS Transit Gateway zur Weiterleitung des Datenverkehrs zwischen der VPC der Lösung und der VPC, die den privaten Endpunkt enthält, in Betracht ziehen. Siehe Transit Gateway Gateway-Dokumentation.
6. Sicherheitsgruppen: Stellen Sie sicher, dass die mit Ihren ECS-Aufgaben verknüpften Sicherheitsgruppen ausgehenden Datenverkehr zum privaten Endpunkt zulassen und dass die Sicherheitsgruppen des privaten Endpunkts eingehenden Datenverkehr von den ECS-Aufgaben zulassen.

Stellen Sie beim Testen interner Application Load Balancers oder EC2-Instances sicher, dass sich die VPC-CIDR-Bereiche nicht überschneiden und dass die erforderlichen Routen in den Routentabellen konfiguriert sind.

Problem: Die Tests werden abgeschlossen, aber die Ergebnisse sind nicht auf der Benutzeroberfläche verfügbar

- Auflösung

Wenn der Test abgeschlossen wurde, die Ergebnisse jedoch nicht in der Benutzeroberfläche verfügbar sind, sollten die Ergebnisdateien weiterhin im S3-Bucket der ECS-Aufgaben verfügbar sein, die die Tests ausgeführt haben. Dies ist eine bekannte Einschränkung der Lösung. In der aktuellen Architektur verwendet die Lösung eine Lambda-Funktion zur Ergebnisanalyse, um die Ergebnisse mehrerer ECS-Aufgaben zusammenzufassen, die dann als Element in der DynamoDB-Tabelle gespeichert werden. Die DynamoDB-Tabelle hat eine maximale Elementgröße von 400 KB. Diese Beschränkung wird abhängig von der Komplexität des Testskripts, der Parallelität und der Anzahl der verwendeten Aufgaben erreicht. Der Fehler bedeutet nicht, dass der Test fehlschlägt, sondern bedeutet, dass der Prozess zur Zusammenfassung der Ergebnisse und deren Speicherung in der DynamoDB-Tabelle für CRUD-Operationen fehlgeschlagen ist. Die Ergebnisse sind weiterhin im S3-Bucket für das Testszenario verfügbar.

Kontaktieren Sie AWS Support.

Wenn Sie über AWS Business Support+, AWS Enterprise Support oder Unified Operations verfügen, können Sie das AWS Support Center nutzen, um fachkundige Support zu dieser Lösung zu erhalten. In den folgenden Abschnitten finden Sie entsprechende Anweisungen.

Fall erstellen

1. Melden Sie sich im Support Center an.
2. Wählen Sie Create case (Fall erstellen) aus.

Wie können wir helfen?

1. Wählen Sie Technisch
2. Wählen Sie für Service die Option Lösungen aus.
3. Wählen Sie für Kategorie die Option Distributed Load Testing on AWS aus.
4. Wählen Sie unter Schweregrad die Option aus, die Ihrem Anwendungsfall am besten entspricht.
5. Wenn Sie den Service, die Kategorie und den Schweregrad eingeben, werden in der Benutzeroberfläche Links zu häufig gestellten Fragen zur Fehlerbehebung angezeigt. Wenn Sie Ihre Fragen mit diesen Links nicht lösen können, wählen Sie Nächster Schritt: Zusätzliche Informationen.

Zusätzliche Informationen

1. Geben Sie als Betreff einen Text ein, der Ihre Frage oder Ihr Problem zusammenfasst.
2. Beschreiben Sie unter Beschreibung das Problem detailliert, einschließlich des Namens dieses Produkts und der Version, die Sie verwenden, z. B. in diesem Beispiel: Distributed Load Testing on AWS vX.Y.Z.
3. Wählen Sie Dateien anhängen.
4. Fügen Sie die Informationen bei, die der AWS-Support zur Bearbeitung der Anfrage benötigt.

Helfen Sie uns, Ihren Fall schneller zu lösen

1. Geben Sie die angeforderten Informationen ein.

2. Klicken Sie auf Next step: Solve now or contact us (()Nächster Schritt): Jetzt lösen oder Support kontaktieren).

Löse es jetzt oder kontaktiere uns

1. Sehen Sie sich die Solve Now-Lösungen an.
2. Wenn Sie Ihr Problem mit diesen Lösungen nicht lösen können, wählen Sie Kontaktieren Sie uns, geben Sie die angeforderten Informationen ein und klicken Sie auf Absenden.

Deinstalliere die Lösung

Sie können die Lösung Distributed Load Testing on AWS über die AWS-Managementkonsole oder über die AWS-Befehlszeilenschnittstelle deinstallieren. Sie müssen die mit dieser Lösung erstellten Konsolen-, Szenario- und Protokollierungs-Buckets von Amazon Simple Storage Service (Amazon S3) manuell löschen. AWS-Lösungsimplementierungen löschen sie nicht automatisch, falls Sie Daten aufbewahren müssen.

> **Note**
>
> Wenn Sie regionale Stacks bereitgestellt haben, müssen Sie die Stacks in diesen Regionen löschen, bevor Sie den Haupt-Stack löschen.

Verwendung der AWS-Managementkonsole

AWS CloudFormation

1. Melden Sie sich bei der CloudFormation AWS-Konsole an.
2. Wählen Sie auf der Seite Stacks den Installations-Stack dieser Lösung aus.
3. Wählen Sie Löschen aus.

AWS-Startassistent

1. Melden Sie sich bei der AWS Launch Wizard Wizard-Konsole an.
2. Wählen Sie auf der Seite Launch Wizard Deployments die Bereitstellung dieser Lösung aus.
3. Wählen Sie Aktionen und dann Löschen aus.
4. Bestätigen Sie das Löschen.

Verwenden der AWS-Befehlszeilenschnittstelle

Stellen Sie fest, ob die AWS-Befehlszeilenschnittstelle (AWS CLI) in Ihrer Umgebung verfügbar ist. Installationsanweisungen finden Sie unter Was ist die AWS-Befehlszeilenschnittstelle im AWS-CLI-Benutzerhandbuch. Nachdem Sie bestätigt haben, dass die AWS-CLI verfügbar ist, führen Sie den folgenden Befehl aus.

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

Löschen der Amazon S3 S3-Buckets

Diese Lösung ist so konfiguriert, dass die von der Lösung erstellten Amazon S3 S3-Buckets (für die Bereitstellung in einer Opt-in-Region) beibehalten werden, falls Sie sich entscheiden, den CloudFormation AWS-Stack zu löschen, um versehentlichen Datenverlust zu verhindern. Nach der Deinstallation der Lösung können Sie diesen S3-Bucket manuell löschen, wenn Sie die Daten nicht behalten müssen. Gehen Sie wie folgt vor, um den Amazon S3 S3-Bucket zu löschen.

1. Melden Sie sich bei der Amazon S3-Konsole an.
2. Wählen Sie im linken Navigationsbereich Buckets aus.
3. Geben Sie im Feld Buckets nach Namen suchen den Namen des Stacks dieser Lösung ein.
4. Wählen Sie einen der S3-Buckets der Lösung aus und wählen Sie Leer.
5. Geben Sie im Bestätigungsfeld den Text Dauerhaft löschen ein und wählen Sie Leer aus.
6. Wählen Sie den S3-Bucket aus, den Sie gerade geleert haben, und wählen Sie Löschen.
7. Geben Sie den S3-Bucket-Namen in das Bestätigungsfeld ein und wählen Sie Bucket löschen.

Wiederholen Sie die Schritte 4 bis 7, bis Sie alle S3-Buckets gelöscht haben.

Führen Sie den folgenden Befehl aus, um den S3-Bucket mithilfe der AWS-CLI zu löschen:

```
$ aws s3 rb s3://<bucket-name> --force
```

Benutze die Lösung

Dieser Abschnitt enthält eine umfassende Anleitung zur Verwendung der Distributed Load Testing on AWS-Lösung, von der Erstellung Ihres ersten Testszenarios bis hin zur Analyse detaillierter Ergebnisse. Der Workflow umfasst die Erstellung eines Testszenarios, die Ausführung eines Tests und die Untersuchung der Testergebnisse.

Erstellen Sie ein Testszenario

Die Erstellung eines Testszenarios umfasst vier Hauptschritte: Konfiguration der allgemeinen Einstellungen, Definition des Szenarios, Gestaltung von Verkehrsmustern und Überprüfung Ihrer Konfiguration.

Schritt 1: Allgemeine Einstellungen

Konfigurieren Sie die grundlegenden Parameter für Ihren Lasttest, einschließlich des Testnamens, der Beschreibung und der allgemeinen Konfigurationsoptionen.

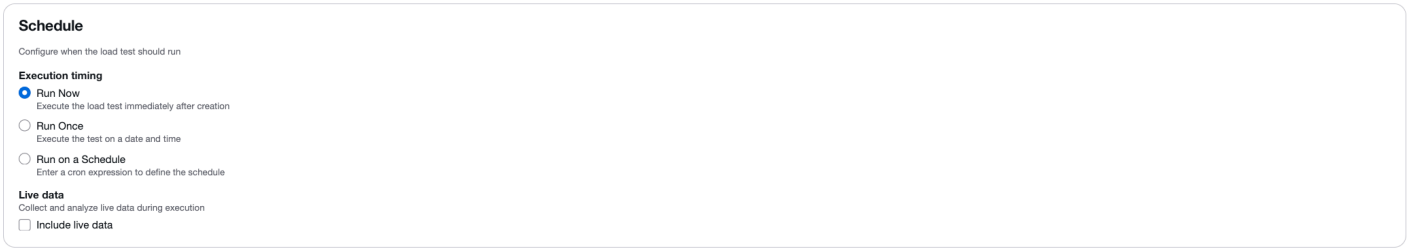
Identifizierung des Tests

- Testname (erforderlich) — Ein beschreibender Name für Ihr Testszenario
- Testbeschreibung (erforderlich) — Zusätzliche Details zum Testzweck und zur Konfiguration
- Tags (optional) — Fügen Sie bis zu 5 Tags hinzu, um Ihre Testszenarien zu kategorisieren und zu organisieren

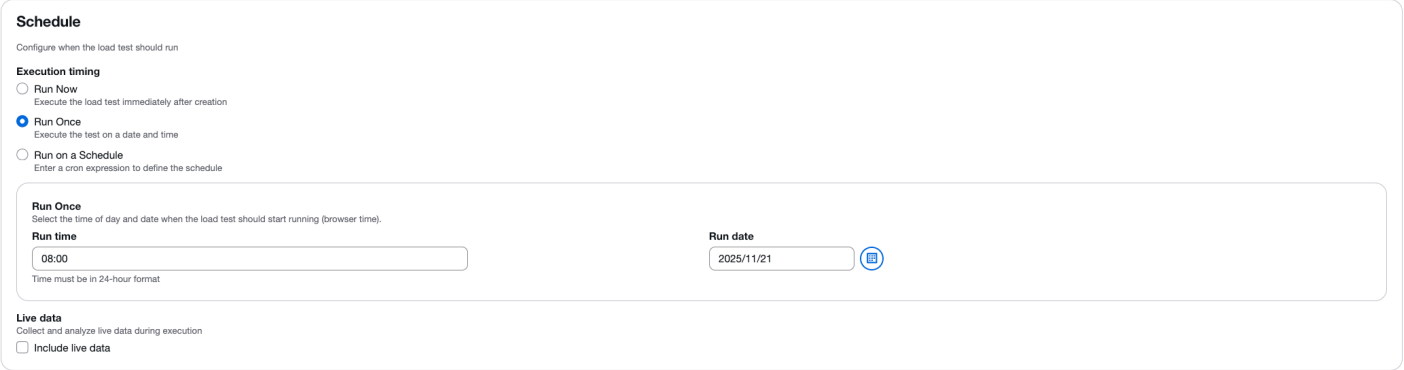
Optionen für die Terminplanung

Konfigurieren Sie, wann der Test ausgeführt werden soll:

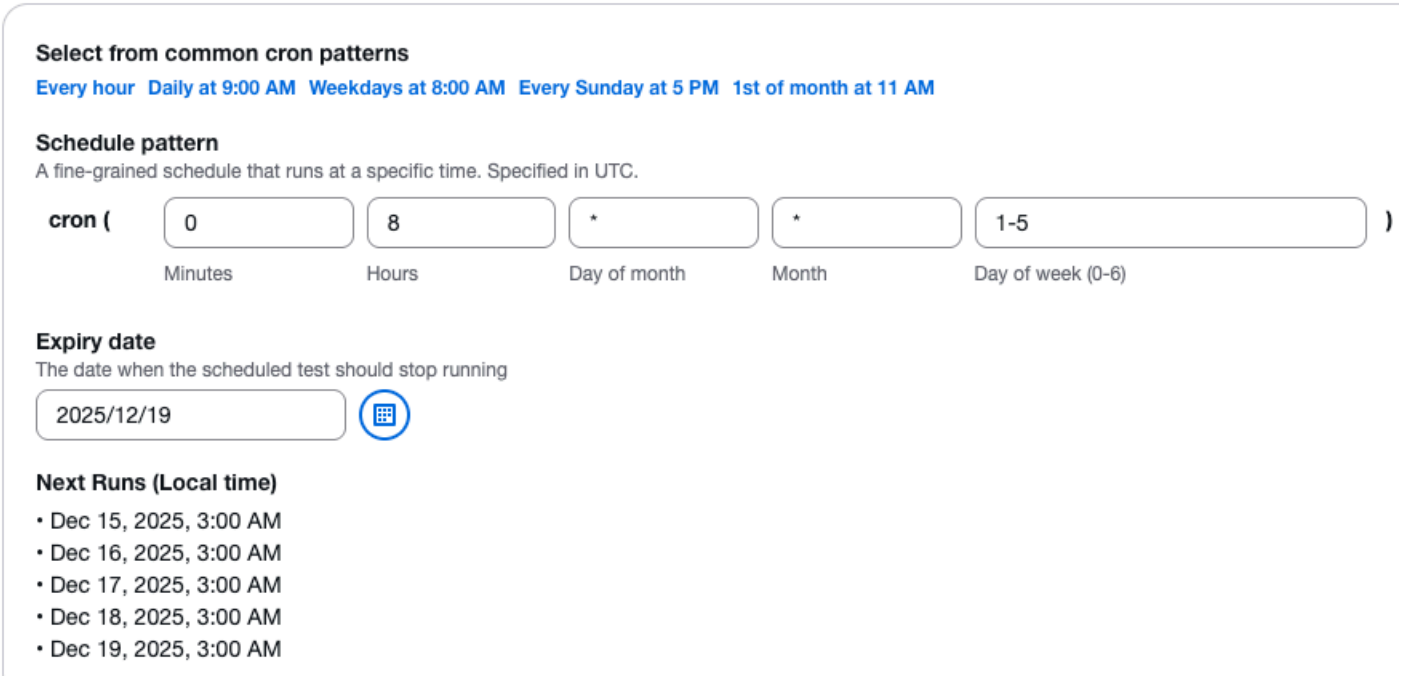
- Jetzt ausführen — Führt den Test unmittelbar nach der Erstellung aus.



- Einmal ausführen — Planen Sie den Test so, dass er an einem bestimmten Datum und zu einer bestimmten Uhrzeit ausgeführt wird.



- Nach einem Zeitplan ausführen — Verwenden Sie die cron-basierte Planung, um Tests automatisch in regelmäßigen Abständen auszuführen. Sie können aus gängigen Mustern (stündlich, täglich, wöchentlich) wählen oder einen benutzerdefinierten Cron-Ausdruck definieren.



Arbeitsablauf planen

Wenn Sie einen Test planen, findet der folgende Workflow statt:

- Die Zeitplanparameter werden über Amazon API Gateway an die API der Lösung gesendet.
- Die API übergibt die Parameter an eine Lambda-Funktion, die eine CloudWatch Event-Regel erstellt, die am angegebenen Datum ausgeführt werden soll.

- Bei einmaligen Tests (Run Once) wird die CloudWatch Events-Regel am angegebenen Datum ausgeführt und die `api-services` Lambda-Funktion führt den Test aus.
- Bei wiederkehrenden Tests (Run on a Schedule) wird die CloudWatch Events-Regel am angegebenen Datum aktiviert, und die `api-services` Lambda-Funktion erstellt eine neue Regel, die sofort und wiederkehrend auf der Grundlage der angegebenen Häufigkeit ausgeführt wird.

Live-Daten

Aktivieren Sie das Kontrollkästchen Live-Daten einbeziehen, um Echtzeitmetriken anzuzeigen, während Ihr Test ausgeführt wird. Wenn diese Option aktiviert ist, können Sie Folgendes überwachen:

- Durchschnittliche Antwortzeit.
- Anzahl virtueller Benutzer.
- Erfolgreiche Anfragen zählen.
- Anzahl fehlgeschlagener Anfragen.

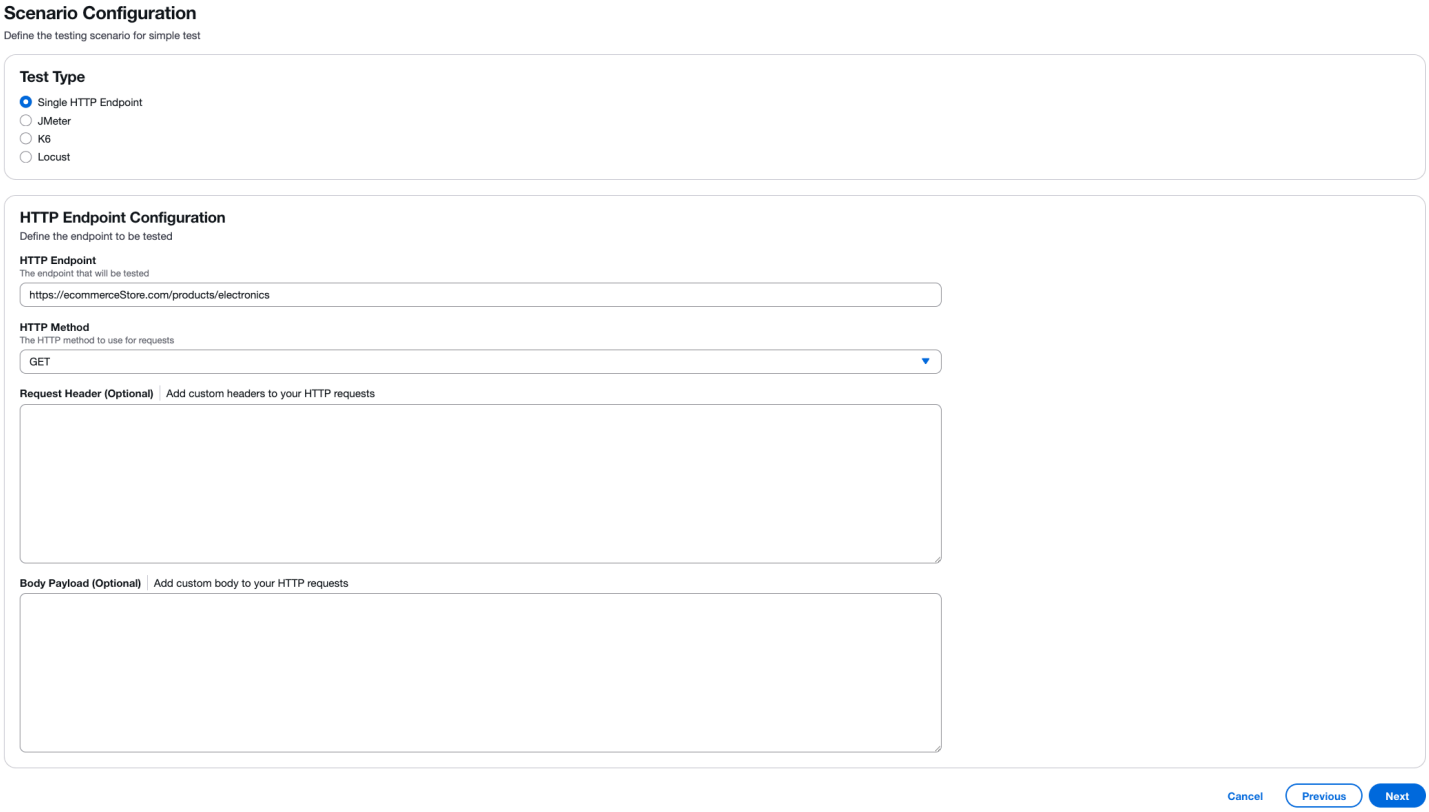
Die Live-Datenfunktion ermöglicht Echtzeitdiagramme mit Daten, die in Intervallen von einer Sekunde aggregiert werden. Weitere Informationen finden Sie unter Überwachung mit Live-Daten.

Schritt 2: Szenariokonfiguration

Definieren Sie das spezifische Testszenario und wählen Sie Ihr bevorzugtes Test-Framework aus.

Auswahl des Testtyps

Wählen Sie die Art des Belastungstests, den Sie durchführen möchten:



- Einzelner HTTP-Endpunkt — Testen Sie einen einzelnen API-Endpunkt oder eine Webseite mit einfacher Konfiguration.
- JMeter- Laden Sie JMeter Testskripte (.jmx-Dateien oder .zip-Archive) hoch.
- K6 — Laden Sie K6-Testskripte (.js-Dateien oder .zip-Archive) hoch.
- Locust - Laden Sie Locust-Testskripte (.py-Dateien oder .zip-Archive) hoch.

HTTP-Endpunktkonfiguration image: :images/test-types.png [Wählen Sie den Testtyp aus, der ausgeführt werden soll] Wenn „Single HTTP Endpoint“ ausgewählt ist, konfigurieren Sie diese Einstellungen:

HTTP-Endpunkt (erforderlich)

Geben Sie die vollständige URL des Endpunkts ein, den Sie testen möchten. Beispiel, `https://api.example.com/users`. Stellen Sie sicher, dass der Endpunkt von der AWS-Infrastruktur aus zugänglich ist.

HTTP-Methode (erforderlich)

Wählen Sie die HTTP-Methode für Ihre Anfragen aus. Der Standardwert ist `GET`. Zu den weiteren Optionen gehören `POST`,`PUT`,`DELETE`,`PATCH`,`HEAD`, und`OPTIONS`.

Header der Anfrage (optional)

Fügen Sie Ihren Anfragen benutzerdefinierte HTTP-Header hinzu. Allgemeine Beispiele sind:

- `Content-Type: application/json`
- `Authorization: Bearer <token>`
- `User-Agent: LoadTest/1.0`

Wählen Sie Header hinzufügen, um mehrere Header einzubeziehen.

Nutzlast des Körpers (optional)

Fügen Sie den Inhalt des Anforderungstexts für POST- oder PUT-Anfragen hinzu. Unterstützt JSON-, XML- oder Nur-Text-Formate. Beispiel: `{"userId": 123, "action": "test"}`.

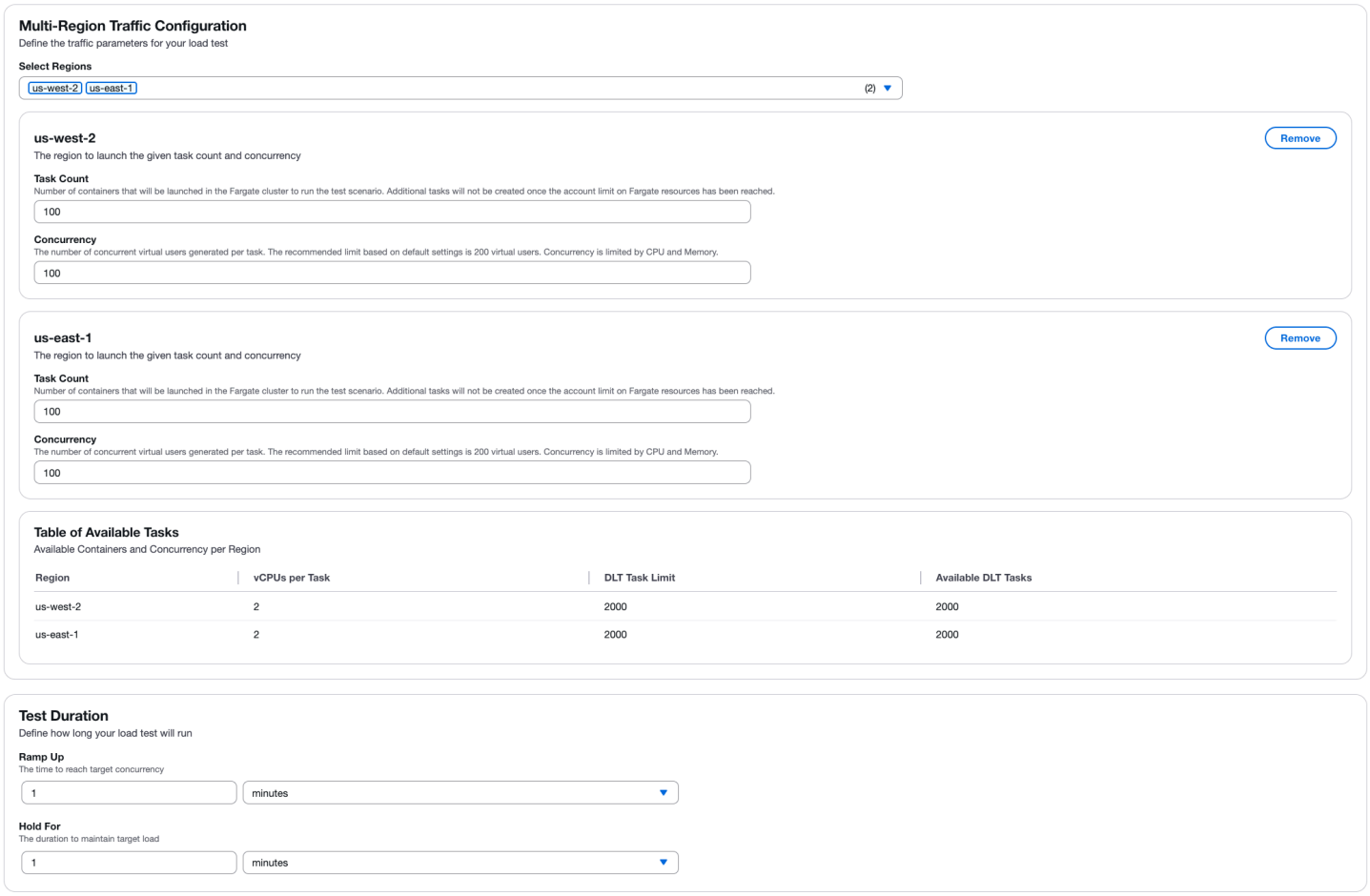
Testen Sie Framework-Skripte

Wenn Sie K6 oder Locust verwenden JMeter, laden Sie Ihre Testskriptdatei oder ein ZIP-Archiv hoch, das Ihr Testskript und unterstützende Dateien enthält. Denn JMeter Sie können benutzerdefinierte Plugins in einen `/plugins` Ordner innerhalb Ihres ZIP-Archivs aufnehmen.

> **Important**
>
> Ihr Testskript (JMeter, K6 oder Locust) kann zwar Parallelität (virtuelle Benutzer), Transaktionsraten (TPS), Anlaufzeiten und andere Ladeparameter definieren, aber die Lösung überschreibt diese Konfigurationen mit den Werten, die Sie während der Testerstellung im Traffic Shape-Bildschirm angeben. Die Traffic Shape-Konfiguration steuert die Anzahl der Aufgaben, die Parallelität (virtuelle Benutzer pro Aufgabe), die Dauer des Hochlaufs und die Haltedauer für die Testausführung.

Schritt 3: Traffic Shape

Konfigurieren Sie, wie der Verkehr während Ihres Tests verteilt werden soll, einschließlich der Unterstützung mehrerer Regionen.



Konfiguration des Datenverkehrs für mehrere Regionen

Wählen Sie eine oder mehrere AWS-Regionen aus, um Ihren Belastungstest geografisch zu verteilen. Konfigurieren Sie für jede ausgewählte Region:

Anzahl der Aufgaben

> Die Anzahl der Container (Aufgaben), die im Fargate-Cluster für das Testszenario gestartet werden. Zusätzliche Aufgaben werden nicht mehr erstellt, sobald das Konto das Limit „Fargate-Ressource wurde erreicht“ erreicht hat.

Concurrency (Nebenläufigkeit)

> Die Anzahl gleichzeitiger virtueller Benutzer, die pro Aufgabe generiert wurden. Das empfohlene Limit basiert auf den Standardeinstellungen von 2 V CPUs pro Aufgabe. Die Parallelität wird durch CPU- und Speicherressourcen begrenzt.

Ermitteln Sie die Anzahl der Benutzer

Die Anzahl der Benutzer, die ein Container für einen Test unterstützen kann, kann bestimmt werden, indem die Anzahl der Benutzer schrittweise erhöht und die Leistung in Amazon überwacht wird CloudWatch. Sobald Sie feststellen, dass die CPU- und Speicherleistung an ihre Grenzen stößt, haben Sie die maximale Anzahl von Benutzern erreicht, die ein Container für diesen Test in seiner Standardkonfiguration unterstützen kann (2 vCPU und 4 GB Arbeitsspeicher).

Der Kalibrierungsprozess

Anhand des folgenden Beispiels können Sie damit beginnen, die Grenzwerte für gleichzeitige Benutzer für Ihren Test zu ermitteln:

1. Erstellen Sie einen Test mit nicht mehr als 200 Benutzern.
2. Überwachen Sie während der Ausführung des Tests die CPU und den Speicher mithilfe der CloudWatch Konsole:
   a. Wählen Sie im linken Navigationsbereich unter Container Insights die Option Performance Monitoring aus.
   b. Wählen Sie auf der Seite zur Leistungsüberwachung im linken Dropdownmenü die Option ECS-Cluster aus.
   c. Wählen Sie im rechten Drop-down-Menü Ihren Amazon Elastic Container Service (Amazon ECS) -Cluster aus.
3. Achten Sie bei der Überwachung auf die CPU und den Arbeitsspeicher. Wenn die CPU nicht über 75% oder der Arbeitsspeicher nicht über 85% liegt (einmalige Spitzenwerte ignorieren), können Sie einen weiteren Test mit einer höheren Benutzerzahl durchführen.

Wiederholen Sie die Schritte 1 bis 3, wenn der Test die Ressourcengrenzen nicht überschritten hat. Optional können Sie die Container-Ressourcen erhöhen, um eine höhere Anzahl gleichzeitiger Benutzer zu ermöglichen. Dies führt jedoch zu höheren Kosten. Einzelheiten finden Sie im Entwicklerhandbuch.

> **Note**
>
> Um genaue Ergebnisse zu erzielen, sollten Sie bei der Festlegung der Grenzwerte für gleichzeitige Benutzer jeweils nur einen Test durchführen. Alle Tests verwenden denselben Cluster, und CloudWatch Container Insights aggregiert die Leistungsdaten auf der Grundlage des Clusters. Dies führt dazu, dass beide Tests gleichzeitig an CloudWatch Container

Insights gemeldet werden, was zu ungenauen Kennzahlen zur Ressourcennutzung für einen einzelnen Test führt.

Weitere Informationen zur Kalibrierung von Benutzern pro Engine finden Sie in der Dokumentation unter Kalibrierung eines Taurus-Tests. BlazeMeter

> **Note**
>
> Die Lösung zeigt Informationen zur verfügbaren Kapazität für jede Region an und hilft Ihnen so, Ihre Testkonfiguration innerhalb der verfügbaren Grenzen zu planen.

Tabelle der verfügbaren Aufgaben

In der Tabelle der verfügbaren Aufgaben wird die Ressourcenverfügbarkeit für jede ausgewählte Region angezeigt:

- Region — Der Name der AWS-Region.
- v CPUs pro Aufgabe — Die Anzahl der virtuellen Geräte, die jeder Aufgabe CPUs zugewiesen sind (Standard: 2).
- DLT-Aufgabenlimit — Die maximale Anzahl von Aufgaben, die basierend auf den Fargate-Limits Ihres Kontos erstellt werden können (Standard: 2000).
- Verfügbare DLT-Aufgaben — Die aktuelle Anzahl von Aufgaben, die in der Region zur Verfügung stehen (Standard: 2000).

Table of Available Tasks
Available Containers and Concurrency per Region

Region	vCPUs per Task	DLT Task Limit	Available DLT Tasks
us-west-2	2	2000	2000
us-east-1	2	2000	2000

Informationen zur Erhöhung der Anzahl der verfügbaren Aufgaben oder v CPUs pro Aufgabe finden Sie im Entwicklerhandbuch.

Dauer des Tests

Definieren Sie, wie lange Ihr Auslastungstest laufen soll:

Hochfahren

Die Zeit bis zum Erreichen der Zielparallelität. Die Auslastung steigt in diesem Zeitraum schrittweise von 0 auf den konfigurierten Parallelitätsgrad.

Halten Sie für

Die Dauer, für die die Ziellast aufrechterhalten werden soll. Der Test wird für diesen Zeitraum mit voller Parallelität fortgesetzt.

Schritt 4: Überprüfen und Erstellen

Überprüfen Sie alle Ihre Konfigurationen, bevor Sie das Testszenario erstellen. Überprüfen:

- Allgemeine Einstellungen (Name, Beschreibung, Zeitplan).
- Szenariokonfiguration (Testtyp, Endpunkt oder Skript).
- Form des Datenverkehrs (Aufgaben, Benutzer, Dauer, Regionen).

Wählen Sie nach der Überprüfung Create aus, um Ihr Testszenario zu speichern.

Testszenarien verwalten

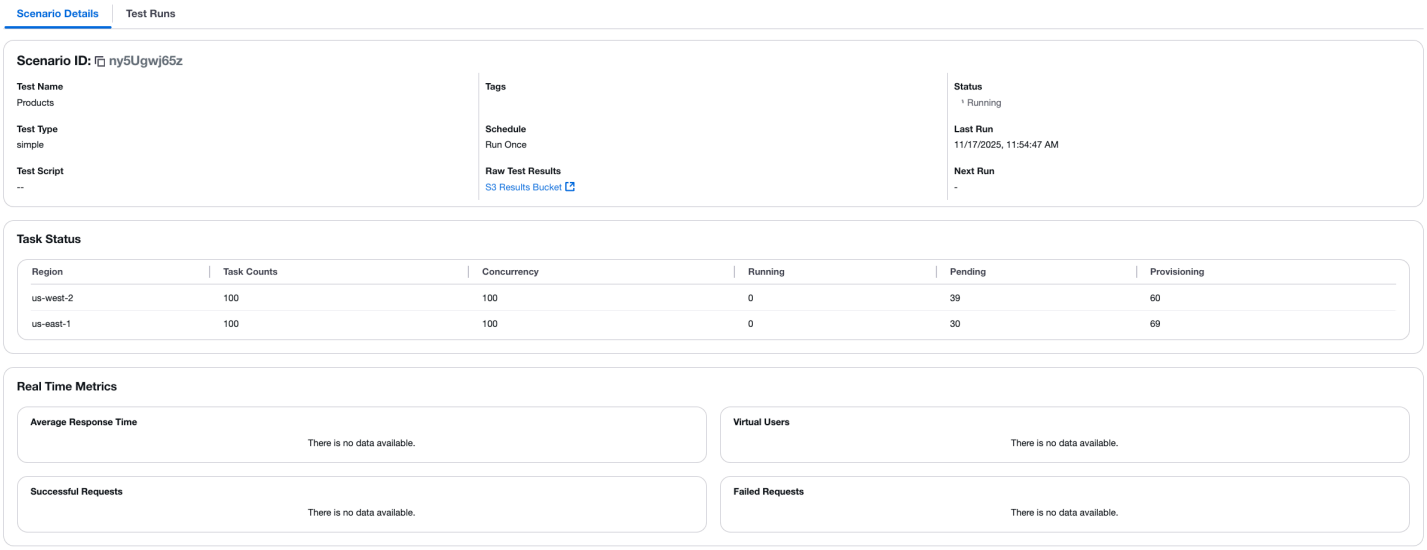
Nachdem Sie ein Testszenario erstellt haben, können Sie:

- Bearbeiten — Ändern Sie die Testkonfiguration. Häufige Anwendungsfälle umfassen:
  - Verfeinerung der Verkehrsform, um die gewünschte Transaktionsrate zu erreichen.
- Kopieren — Duplizieren Sie ein vorhandenes Testszenario, um Varianten zu erstellen. Häufige Anwendungsfälle umfassen:
  - Endpunkte aktualisieren oder headers/body Parameter hinzufügen.
  - Hinzufügen oder Ändern von Testskripten.
- Löschen — Entfernen Sie Testszenarien, die Sie nicht mehr benötigen.

Führen Sie ein Testszenario aus

Nachdem Sie ein Testszenario erstellt haben, können Sie es sofort ausführen oder so planen, dass es zu einem bestimmten Zeitpunkt in der future ausgeführt wird. Wenn Sie zu einem laufenden Test

navigieren, zeigt die Konsole die Registerkarte Szenariodetails mit Aufgabenstatus und Messdaten in Echtzeit an.



Ansicht der Szenariodetails

Auf der Registerkarte Szenariodetails werden wichtige Informationen zu Ihrem Test angezeigt. In der Tabelle mit dem Aufgabenstatus werden Echtzeitinformationen für jede Region angezeigt.

Tabelle mit dem Aufgabenstatus

In der Tabelle mit dem Aufgabenstatus werden Echtzeitinformationen für jede Region angezeigt:

- Region — Die AWS-Region, in der Aufgaben ausgeführt werden
- Anzahl der Aufgaben — Die Gesamtzahl der für die Region konfigurierten Aufgaben
- Parallelität — Die Anzahl der virtuellen Benutzer pro Aufgabe
- Wird ausgeführt — Anzahl der Aufgaben, die den Test derzeit ausführen
- Ausstehend — Anzahl der Aufgaben, die darauf warten, gestartet zu werden
- Bereitstellung — Anzahl der Aufgaben, die bereitgestellt werden

Arbeitsablauf bei der Testausführung

Wenn ein Test gestartet wird, findet der folgende Workflow statt:

1. Aufgabenbereitstellung — Die Lösung stellt Container (Aufgaben) in den angegebenen AWS-Regionen bereit. Aufgaben werden in der Spalte „Bereitstellung“ angezeigt.

2. Aufgabenstart — Die Lösung stellt weiterhin Aufgaben bereit, bis die Zielanzahl der Aufgaben in jeder Region erreicht ist. Aufgaben werden von „Bereitstellung“ zu „Ausstehend“ zu „Wird ausgeführt“ verschoben.
3. Generierung von Traffic — Nachdem die Lösung alle Aufgaben in einer Region bereitgestellt hat, beginnt sie, Traffic an Ihren Zielendpunkt zu senden.
4. Testausführung — Der Test wird für die konfigurierte Dauer (Hochlauf + Haltezeit) ausgeführt.
5. Analyse der Ergebnisse — Wenn der Test beendet ist, aggregiert und verarbeitet ein Hintergrundanalysejob die Ergebnisse aus allen Regionen.

Status des Testlaufs

Testläufe können die folgenden Status haben:

- Geplant — Der Test soll in der future ausgeführt werden.
- Wird ausgeführt — Der Test ist derzeit im Gange.
- Abgebrochen — Ein Benutzer hat einen laufenden Testlauf abgebrochen.
- Fehlerhaft — Beim Testlauf ist ein Fehler aufgetreten.
- Abgeschlossen — Der Testlauf wurde erfolgreich abgeschlossen und die Ergebnisse liegen vor.

Überwachung mit Live-Daten

Wenn Sie bei der Erstellung des Testszenarios Live-Daten aktiviert haben, können Sie Echtzeitmetriken anzeigen, während der Test ausgeführt wird. Im Abschnitt Echtzeit-Metriken werden vier Diagramme angezeigt, die im Verlauf des Tests kontinuierlich aktualisiert werden. Die Daten werden in Intervallen von einer Sekunde aggregiert.

Beschreibungen der Grafiken

Durchschnittliche Antwortzeit

Zeigt die durchschnittliche Antwortzeit in Sekunden für Anfragen an, die von jeder Region verarbeitet wurden. Die Y-Achse zeigt die Antwortzeit in Sekunden und die X-Achse die Tageszeit. Jede Region wird in der Legende durch eine andere Farbe dargestellt.

Virtuelle Benutzer

Zeigt die Anzahl der gleichzeitigen virtuellen Benutzer an, die in jeder Region aktiv Last erzeugen. Das Diagramm zeigt, wie virtuelle Benutzer während des Tests an Wert gewinnen und das angestrebte Parallelitätsniveau beibehalten wird.

Erfolgreiche Anfragen

Zeigt die Gesamtanzahl erfolgreicher Anfragen im Laufe der Zeit für jede Region an. Das Diagramm zeigt die Geschwindigkeit, mit der erfolgreiche Anfragen verarbeitet werden.

Fehlgeschlagene Anfragen

Zeigt die Gesamtzahl der fehlgeschlagenen Anfragen im Laufe der Zeit für jede Region an. Eine niedrige Anzahl oder Null weist auf eine fehlerfreie Testausführung hin.

Visualisierung mehrerer Regionen

Wenn Tests in mehreren Regionen ausgeführt werden, zeigt jedes Diagramm Daten für alle Regionen gleichzeitig an. Die Legende am unteren Rand jedes Diagramms gibt an, welche Farbe die einzelnen Regionen repräsentiert (z. B. us-west-2 und us-east-1).

Technische Umsetzung

Die CloudWatch Protokollgruppe für die Fargate-Aufgaben enthält einen Abonnementfilter, der Testergebnisse erfasst. Wenn das Muster erkannt wird, strukturiert eine Lambda-Funktion die Daten und veröffentlicht sie in einem AWS IoT Core Core-Thema. Die Webkonsole abonniert dieses Thema und zeigt die Metriken in Echtzeit an.

> **Note**
>
> Live-Daten sind kurzlebig und nur verfügbar, während der Test läuft. Die Webkonsole speichert maximal 5.000 Datenpunkte. Danach werden die ältesten Daten durch die neuesten ersetzt. Wenn die Seite aktualisiert wird, sind die Grafiken leer und beginnen mit dem nächsten verfügbaren Datenpunkt. Sobald ein Test abgeschlossen ist, speichert die Lösung die Ergebnisdaten in DynamoDB und Amazon S3. Wenn noch keine Daten verfügbar sind, wird in den Diagrammen „Es sind keine Daten verfügbar“ angezeigt.

Einen Test stornieren

Sie können einen laufenden Test über die Webkonsole abbrechen. Wenn Sie einen Test abbrechen, läuft der folgende Arbeitsablauf ab:

1. Die Stornierungsanfrage wird an die `microservices` API gesendet
2. Die `microservices` API ruft die `task-canceler` Lambda-Funktion auf, die alle aktuell gestarteten Aufgaben stoppt.
3. Wenn die `task-runner` Lambda-Funktion nach dem ersten Abbruchaufruf weiter ausgeführt wird, werden Aufgaben möglicherweise weiterhin kurz gestartet.
4. Sobald die `task-runner` Lambda-Funktion abgeschlossen ist, fährt AWS Step Functions mit dem `Cancel Test` Schritt fort, in dem die `task-canceler` Lambda-Funktion erneut ausgeführt wird, um alle verbleibenden Aufgaben zu beenden

> **Note**
>
> Bei abgebrochenen Tests dauert es einige Zeit, bis der Shutdown-Vorgang abgeschlossen ist, da die Lösung alle Container beendet. Der Teststatus ändert sich auf „Storniert“, sobald alle Ressourcen bereinigt sind.

Erkunden Sie die Testergebnisse

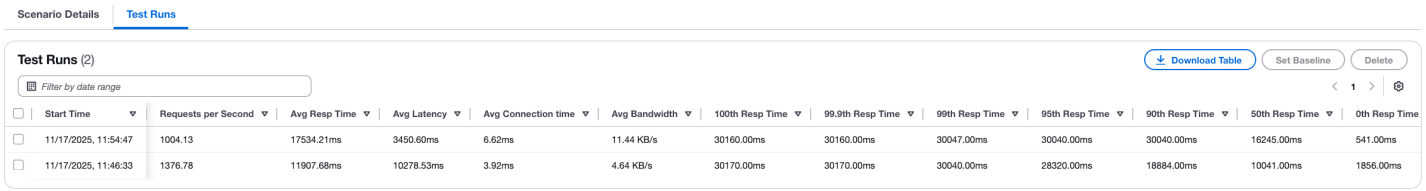
Sobald der Parsing-Job abgeschlossen ist, stehen die Testergebnisse zur Analyse zur Verfügung. Die Lösung bietet umfassende Metriken und Tools, die Ihnen helfen, die Leistung Ihrer Anwendung unter Last zu verstehen.

Übersichtsmetriken für den Testlauf

Wenn ein Test abgeschlossen ist, generiert die Lösung eine Zusammenfassung, die die folgenden Kennzahlen enthält:

- Durchschnittliche Antwortzeit — Die durchschnittliche Antwortzeit in Sekunden für alle durch den Test generierten Anfragen.
- Durchschnittliche Latenz — Die durchschnittliche Latenz in Sekunden für alle durch den Test generierten Anfragen.
- Durchschnittliche Verbindungszeit — Die durchschnittliche Zeit in Sekunden, die für alle Anfragen benötigt wird, um eine Verbindung zum Host herzustellen.
- Durchschnittliche Bandbreite — Die durchschnittliche Bandbreite für alle durch den Test generierten Anfragen.
- Gesamtzahl — Die Gesamtzahl der Anfragen.
- Anzahl erfolgreicher Anfragen — Die Gesamtzahl der erfolgreichen Anfragen.
- Anzahl der Fehler — Die Gesamtzahl der Fehler.
- Anfragen pro Sekunde — Die durchschnittlichen Anfragen pro Sekunde für alle vom Test generierten Anfragen.
- Perzentile — Perzentile für die Antwortzeit, einschließlich p50 (Median), p90, p95 und p99, die die Verteilung der Antwortzeiten auf alle Anfragen zeigen.

Tabelle mit Testläufen



In der Tabelle mit den Testläufen werden alle historischen Testläufe für ein Szenario angezeigt. Sie können:

- Übersichtsmetriken für jeden Testlauf anzeigen.
- Legen Sie einen Basistestlauf für den Leistungsvergleich fest.
- Laden Sie die Tabelle als CSV-Datei herunter.
- Wechseln Sie zwischen den Spalten, um Ihre Ansicht anzupassen.
- Wählen Sie einen Testlauf aus, um detaillierte Ergebnisse anzuzeigen.

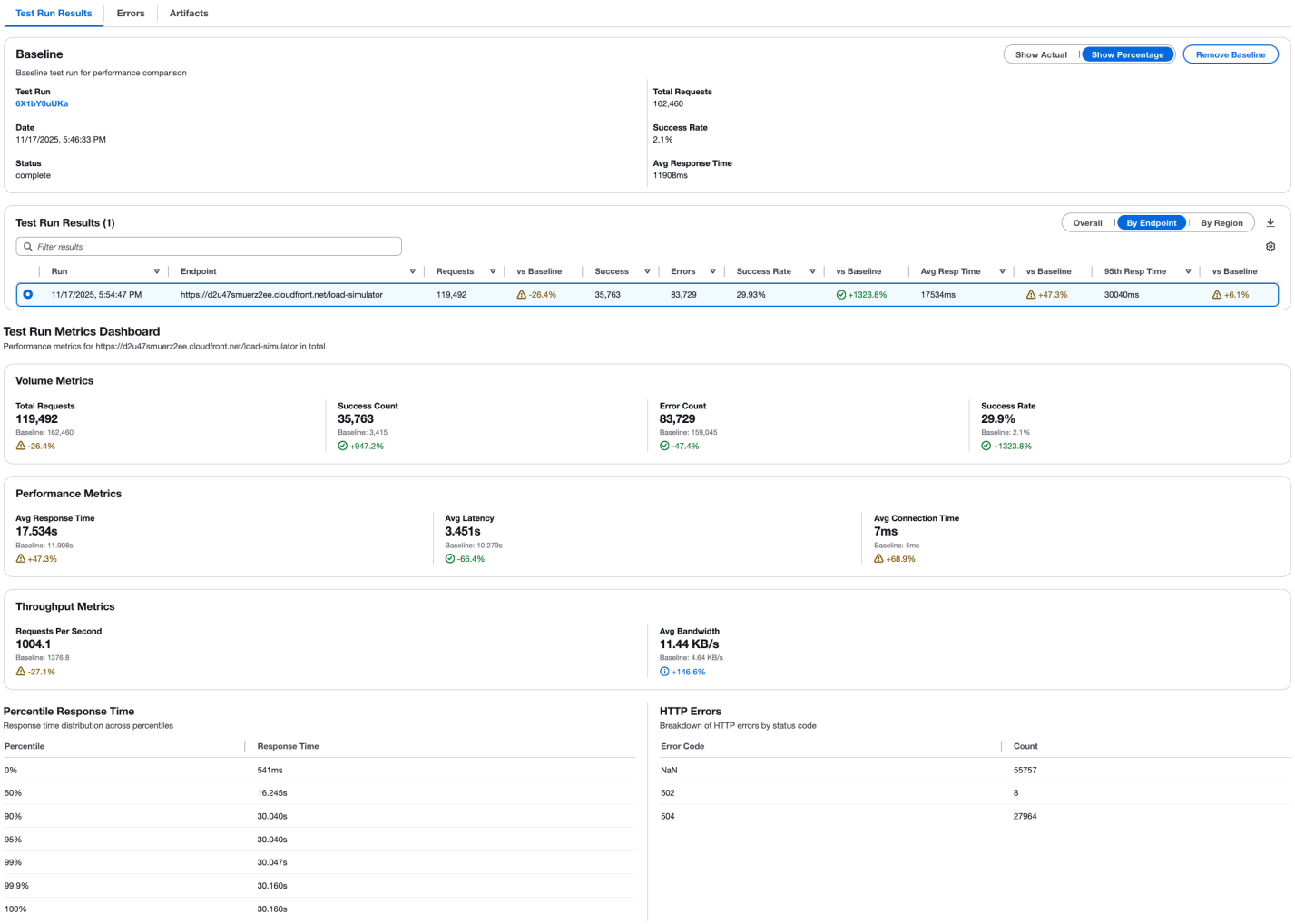
Vergleich der Ausgangswerte

Sie können einen Testlauf als Ausgangswert festlegen, um future Testläufe damit zu vergleichen. Wenn ein Basiswert festgelegt ist:

- Die Tabelle mit den Testläufen zeigt für jede Metrik prozentuale Unterschiede (+/%) im Vergleich zum Ausgangswert.
- Der Basisindikator hilft Ihnen dabei, Leistungsverbesserungen oder Regressionen schnell zu erkennen.
- Sie können den Basiswert jederzeit ändern oder löschen.

Detaillierte Testergebnisse

Wenn Sie einen Testlauf auswählen, wird die detaillierte Ergebnisansicht mit drei Registerkarten geöffnet: Testlaufergebnisse, Fehler und Artefakte.



Basisinformationen

Wenn ein Basistestlauf festgelegt ist, wird dieser oben auf der Seite angezeigt. Sie können „Aktuellen Wert anzeigen“, „Prozentsatz anzeigen“ oder „Basiswert entfernen“ wählen, um zu steuern, wie Basisvergleiche angezeigt werden.

Tabelle mit den Ergebnissen des Testlaufs

Die Ergebnistabelle enthält detaillierte Messwerte mit den folgenden Funktionen:

Dimensionsansichten

Wechseln Sie mit den Bemaßungsschaltflächen zwischen drei Ansichten:

- Insgesamt — Aggregierte Ergebnisse für alle Endpunkte und Regionen
- Nach Endpunkten — Die Ergebnisse sind nach einzelnen Endpunkten aufgeschlüsselt

- Nach Regionen — Ergebnisse aufgeschlüsselt nach AWS-Regionen

Aktionsschaltflächen

- Aktuelle Werte anzeigen — Zeigt die tatsächlichen Metrikwerte an
- Prozentsatz anzeigen — Zeigt prozentuale Abweichungen vom Ausgangswert an
- Basisplan entfernen — Löscht den Basisvergleich

Datenexport und Anpassung

- Laden Sie die Ergebnistabelle als CSV-Datei herunter
- Wechseln Sie zwischen den Spalten, um Ihre Ansicht anzupassen
- Filtern und sortieren Sie Daten, um sich auf bestimmte Kennzahlen zu konzentrieren
- Filtern und sortieren Sie Daten, um sich auf bestimmte Kennzahlen zu konzentrieren.

Registerkarte „Fehler“

Die Registerkarte Fehler bietet eine detaillierte Fehleranalyse:

- Zeigt die Anzahl der Fehler nach Typ an.
- Lassen Sie sich die Fehler nach Gesamttest oder Endpunkt aggregiert anzeigen.
- Identifizieren Sie Muster bei fehlgeschlagenen Anfragen.
- Beheben Sie Probleme mit bestimmten Endpunkten oder Regionen.

Registerkarte „Artefakte“

Über die Registerkarte Artefakte können Sie auf alle Dateien zugreifen, die während des Testlaufs generiert wurden:

- Einzelne Artefakte (Protokolle, Ergebnisdateien) anzeigen.
- Laden Sie bestimmte Artefakte für die Offline-Analyse herunter.
- Laden Sie alle Testlauf-Artefakte als ein einziges Archiv herunter.

Struktur der S3-Ergebnisse

In Version 4.0 wurde die S3-Ergebnisstruktur geändert, um die Organisation zu verbessern:

- Neue Struktur -`scenario-id/test-run-id/results-files`.
- Legacy-Struktur — Tests, die vor Version 4.0 ausgeführt wurden, zeigen alle Ergebnisdateien auf Szenario-ID-Ebene.

> **Note**
>
> Die Testergebnisse werden in der Konsole angezeigt. Sie können auch direkt im Amazon S3 S3-Bucket unter dem `Results` Ordner auf die Rohtestergebnisse zugreifen. Weitere Informationen zu den Taurus-Testergebnissen finden Sie unter Generieren von Testberichten im Taurus-Benutzerhandbuch.

MCP-Serverintegration

Wenn Sie die optionale MCP-Serverkomponente während der Lösungsbereitstellung bereitgestellt haben, können Sie die Distributed Load Testing-Lösung in KI-Entwicklungstools integrieren, die das Model Context Protocol unterstützen. Der MCP-Server bietet programmatischen Zugriff zum Abrufen, Verwalten und Analysieren von Lasttests mithilfe von KI-Assistenten.

Kunden können mit dem Client ihrer Wahl (Amazon Q, Claude usw.) eine Verbindung zum DLT MCP-Server herstellen, für die jeweils leicht unterschiedliche Konfigurationsanweisungen gelten. Dieser Abschnitt enthält Anweisungen zur Einrichtung von MCP Inspector, Amazon Q CLI, Cline und Amazon Q Suite.

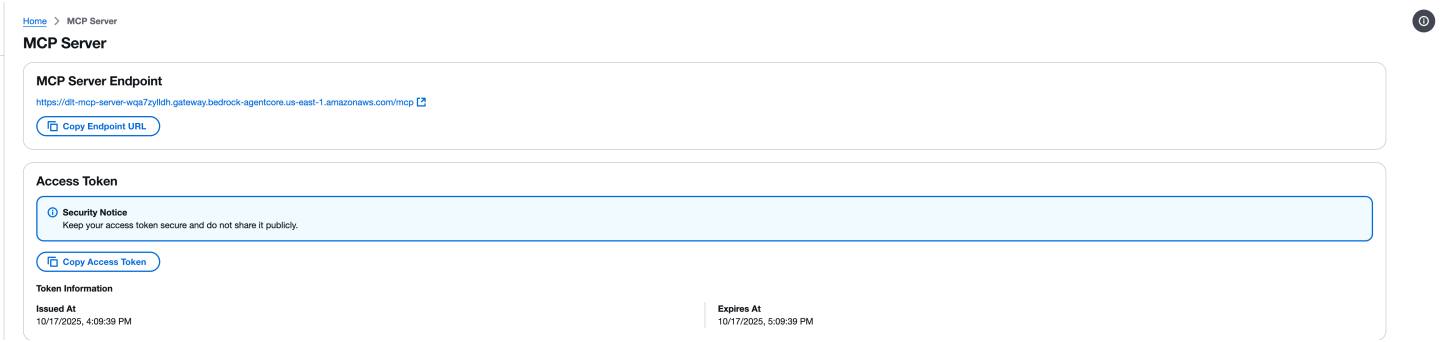
Schritt 1: Holen Sie sich den MCP-Endpunkt und das Zugriffstoken

Bevor Sie einen MCP-Client konfigurieren, müssen Sie Ihren MCP-Serverendpunkt und Ihr Zugriffstoken von der DLT-Webkonsole abrufen.

1. Navigieren Sie in der Distributed Load Testing-Webkonsole zur MCP-Serverseite.
2. Suchen Sie den Abschnitt MCP Server Endpoint.
3. Kopieren Sie die Endpunkt-URL mithilfe der Schaltfläche Endpunkt-URL kopieren. Die Endpunkt-URL folgt dem Format: `https://{gateway-id}.gateway.bedrock-agentcore.{region}.amazonaws.com/mcp`
4. Suchen Sie den Abschnitt Access Token.
5. Kopieren Sie das Zugriffstoken mit der Schaltfläche Zugriffstoken kopieren.

Important

Bewahren Sie Ihr Zugriffstoken sicher auf und geben Sie es nicht öffentlich weiter. Das Token bietet über die MCP-Schnittstelle schreibgeschützten Zugriff auf Ihre Distributed Load Testing-Lösung.



Schritt 2: Testen Sie mit MCP Inspector

Das Model Context Protocol bietet MCP Inspector, ein Tool zum direkten Herstellen einer Verbindung zu MCP-Servern und zum Aufrufen von Tools. Dies bietet eine praktische Benutzeroberfläche und Beispielnetzwerkanforderungen zum Testen Ihrer MCP-Serververbindung vor der Konfiguration von AI-Clients.

Note

MCP Inspector benötigt Version 0.17 oder höher. Alle Anfragen können auch direkt mit JSON RPC gestellt werden, MCP Inspector bietet jedoch eine benutzerfreundlichere Oberfläche.

Installieren und starten Sie MCP Inspector

1. Installieren Sie npm, falls erforderlich.
2. Führen Sie den folgenden Befehl aus, um MCP Inspector zu starten:

```
npx @modelcontextprotocol/inspector
```

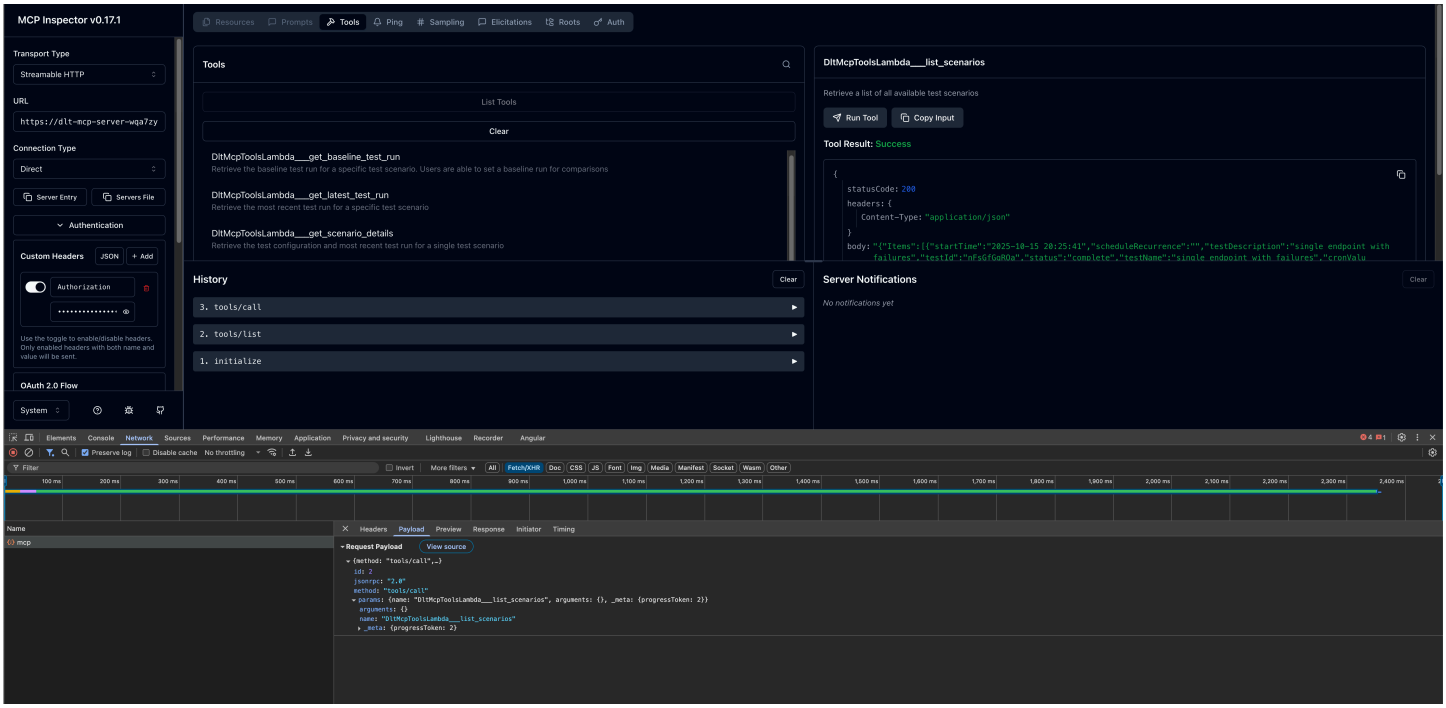
Konfigurieren Sie die Verbindung

1. Geben Sie in der MCP Inspector-Oberfläche Ihre MCP Server-Endpunkt-URL ein.
2. Fügen Sie einen Autorisierungsheader mit Ihrem Zugriffstoken hinzu.
3. Klicken Sie auf Connect, um die Verbindung herzustellen.

Rufen Sie Tools auf

Sobald die Verbindung hergestellt ist, können Sie die verfügbaren MCP-Tools testen:

1. Durchsuchen Sie die Liste der verfügbaren Tools im linken Bereich.
2. Wählen Sie ein Werkzeug aus (z. B.`list_scenarios`).
3. Geben Sie alle erforderlichen Parameter an.
4. Klicken Sie auf Aufrufen, um das Tool auszuführen und die Antwort anzuzeigen.



Schritt 3: Konfigurieren Sie KI-Entwicklungsclients

Nachdem Sie Ihre MCP-Serververbindung mit MCP Inspector überprüft haben, können Sie Ihren bevorzugten AI-Entwicklungsclient konfigurieren.

Amazon Q CLI

Amazon Q CLI bietet Befehlszeilenzugriff auf KI-gestützte Entwicklung mit MCP-Serverintegration.

Schritte zur Konfiguration

1. Bearbeiten Sie die `mcp.json` Konfigurationsdatei. Weitere Informationen zum Speicherort der Konfigurationsdatei finden Sie unter Konfiguration von Remote-MCP-Servern im Amazon Q Developer User Guide.
2. Fügen Sie Ihre DLT-MCP-Serverkonfiguration hinzu:

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "http",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
```

```
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

Überprüfen Sie die Konfiguration

1. Geben Sie in einem Terminal ein, q um Amazon Q CLI zu starten.
2. Geben Sie `/mcp` ein, um alle verfügbaren MCP-Server zu sehen.
3. Geben Sie `/tools` ein, um die verfügbaren Tools von `dlt-mcp` und anderen konfigurierten MCP-Servern anzuzeigen.
4. Stellen Sie sicher, dass die Initialisierung `dlt-mcp` erfolgreich war.

Cline

Cline ist ein KI-Codierungsassistent, der die MCP-Serverintegration unterstützt.

Schritte zur Konfiguration

1. Navigieren Sie in Cline zu MCP-Server verwalten > Konfigurieren > MCP-Server konfigurieren.
2. Aktualisieren Sie die Datei: `cline_mcp_settings.json`

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "streamableHttp",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

3. Speichern Sie die Konfigurationsdatei.
4. Starten Sie Cline neu, um die Änderungen zu übernehmen.

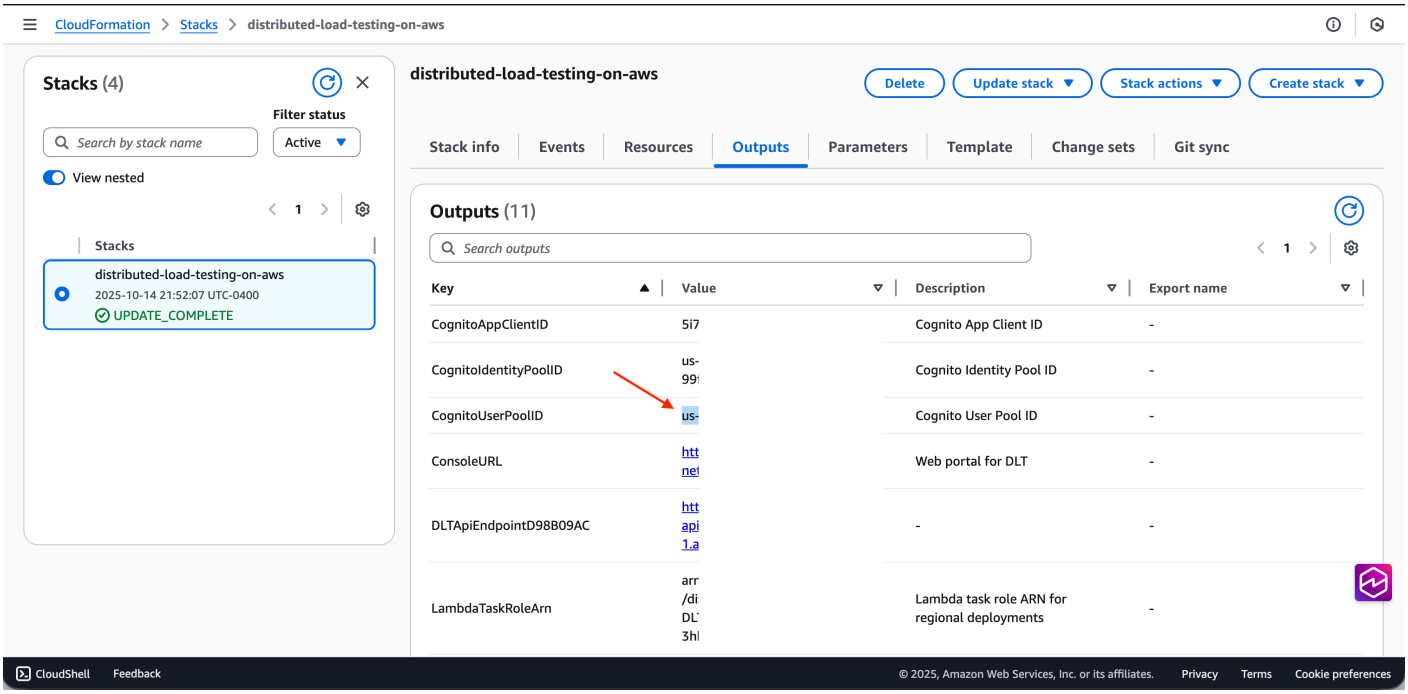
Amazon Q Suite

Amazon Q Suite bietet eine umfassende KI-Assistentenplattform mit Unterstützung für MCP-Serveraktionen.

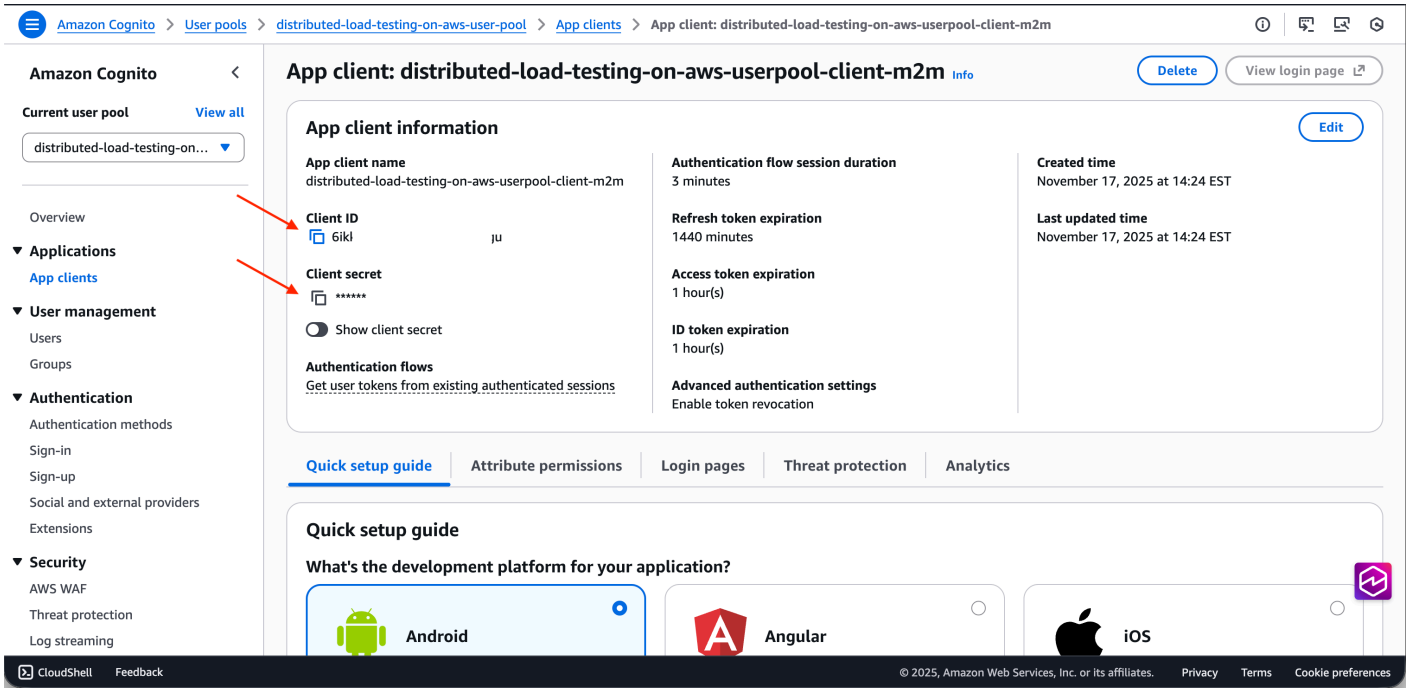
Voraussetzungen

Bevor Sie den MCP-Server in Amazon Q Suite konfigurieren, müssen Sie OAuth Anmeldeinformationen aus dem Cognito-Benutzerpool Ihrer DLT-Bereitstellung abrufen:

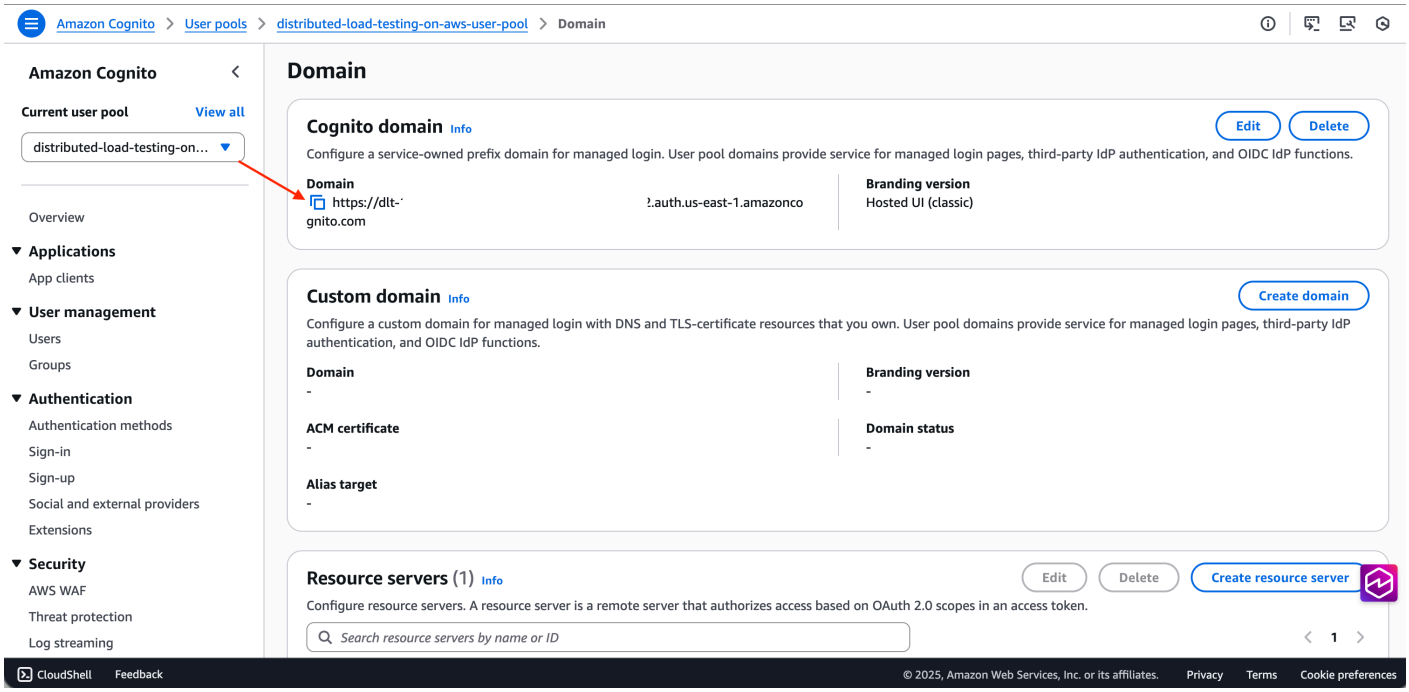
1. Navigieren Sie zur CloudFormation AWS-Konsole.
2. Wählen Sie den Distributed Load Testing-Stack aus.
3. Suchen Sie auf der Registerkarte Ausgaben die Cognito-Benutzerpool-ID, die Ihrer DLT-Bereitstellung zugeordnet ist, und kopieren Sie sie.



4. Navigieren Sie zur Amazon-Cognito-Konsole.
5. Wählen Sie den Benutzerpool anhand der Benutzerpool-ID aus den CloudFormation Ausgaben aus.
6. Wählen Sie in der linken Navigationsleiste App-Integration > App-Clients aus.



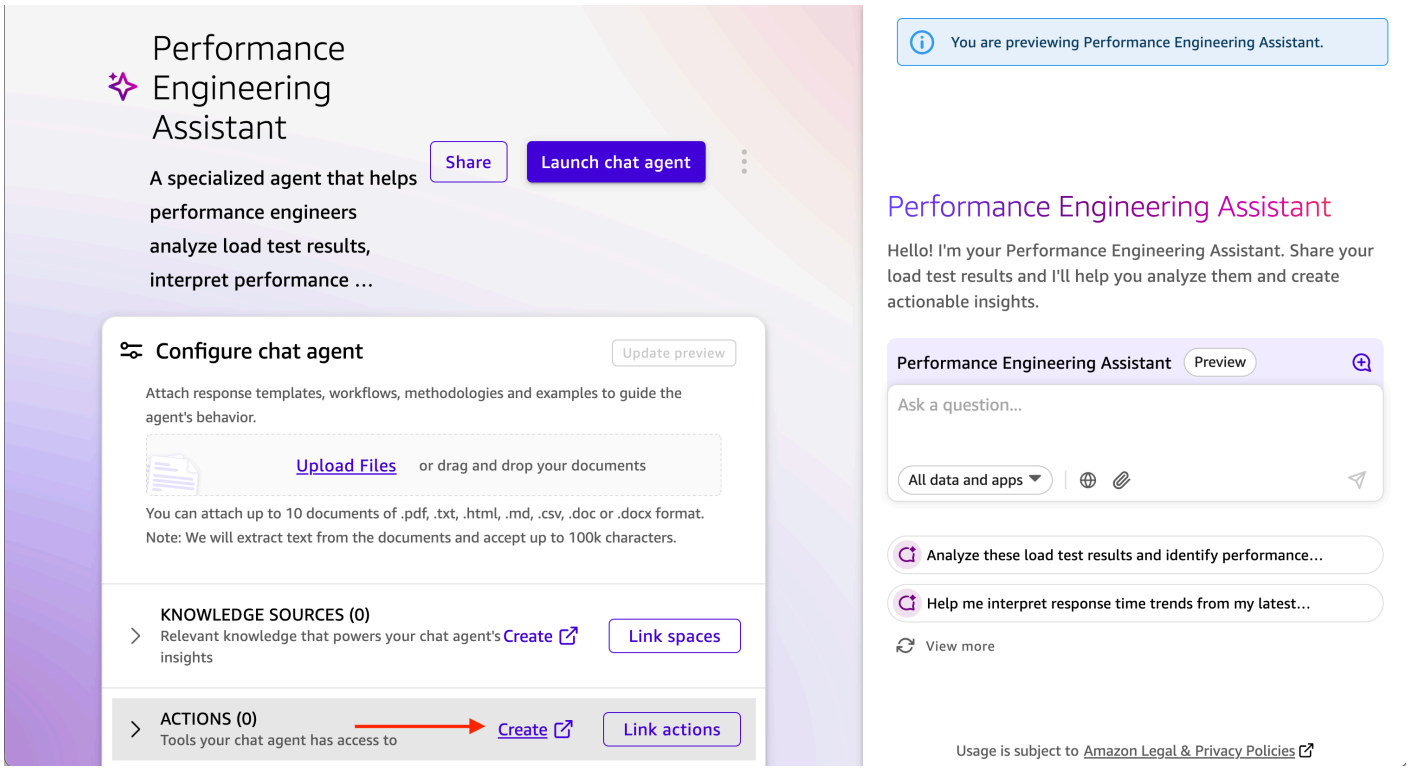
7. Suchen Sie den App-Client, dessen Name auf `m2m` (machine-to-machine) endet.
8. Kopieren Sie die Client-ID und den geheimen Client-Schlüssel.
9. Rufen Sie die Benutzerpool-Domain auf der Registerkarte Domäne ab.

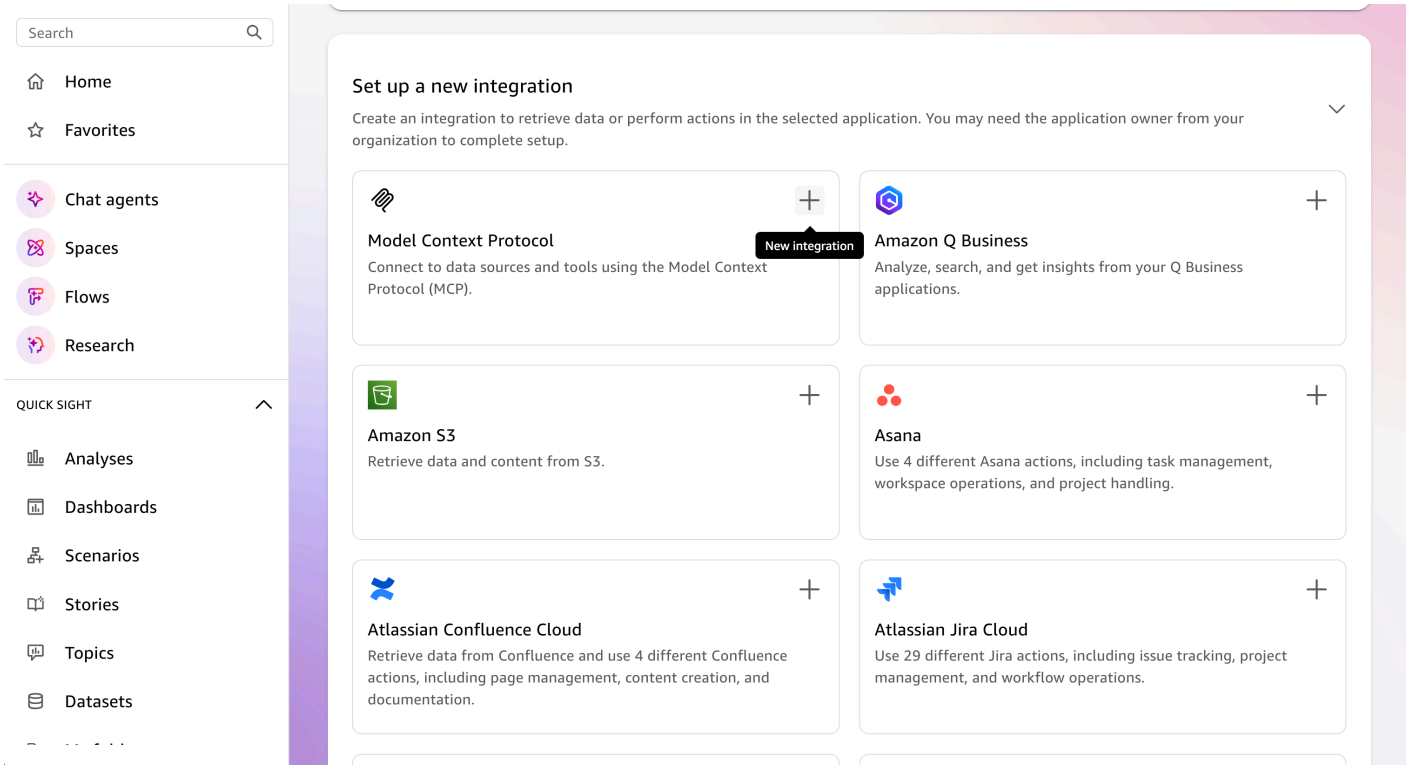


10. Erstellen Sie die Token-Endpunkt-URL, indem `/oauth2/token` Sie sie an das Ende der Domain anhängen.

Schritte zur Konfiguration

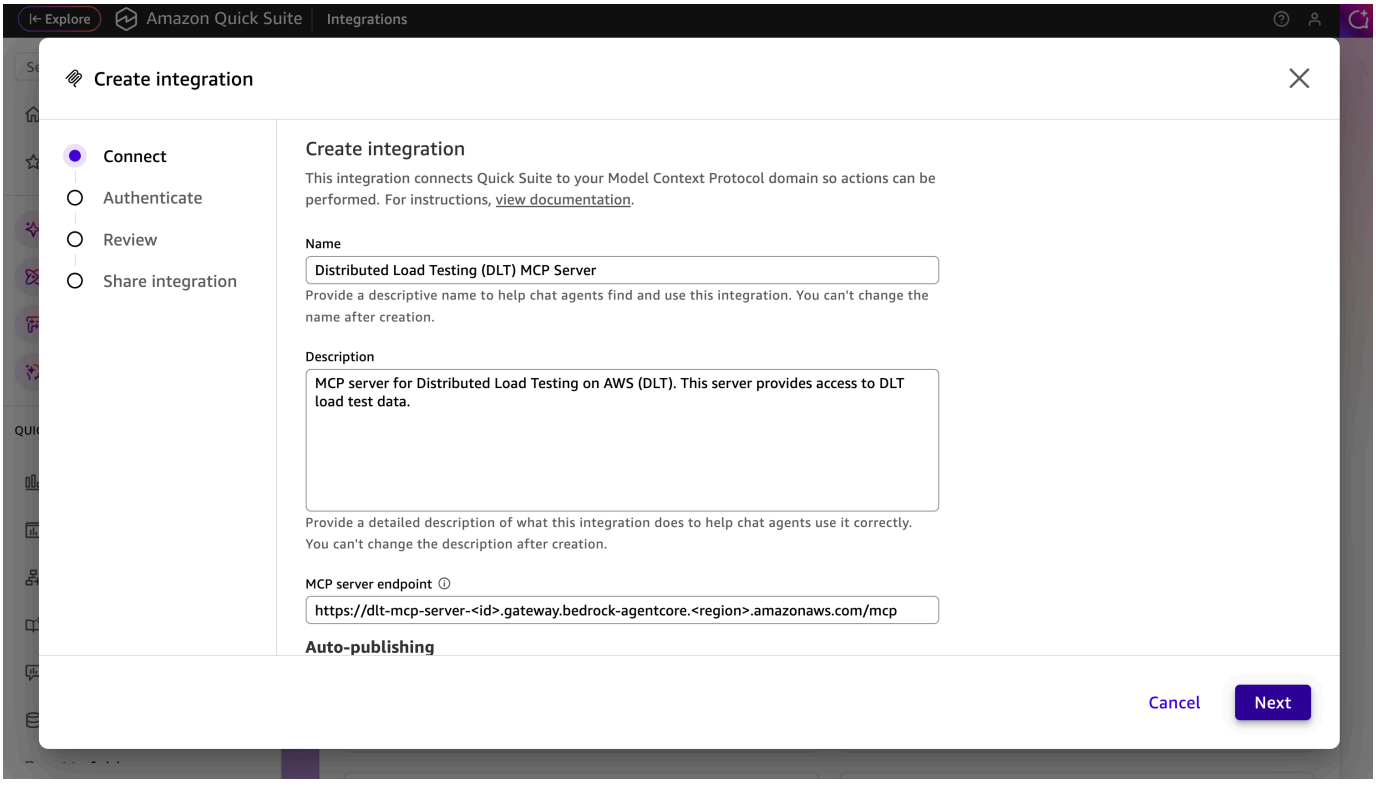
1. Erstellen Sie in Amazon Q Suite einen neuen Agenten oder wählen Sie einen vorhandenen Agenten aus.
2. Fügen Sie eine Agenten-Eingabeaufforderung hinzu, die beschreibt, wie Sie mit dem DLT MCP-Server interagieren.
3. Fügen Sie eine neue Aktion hinzu und wählen Sie MCP-Serveraktion aus.





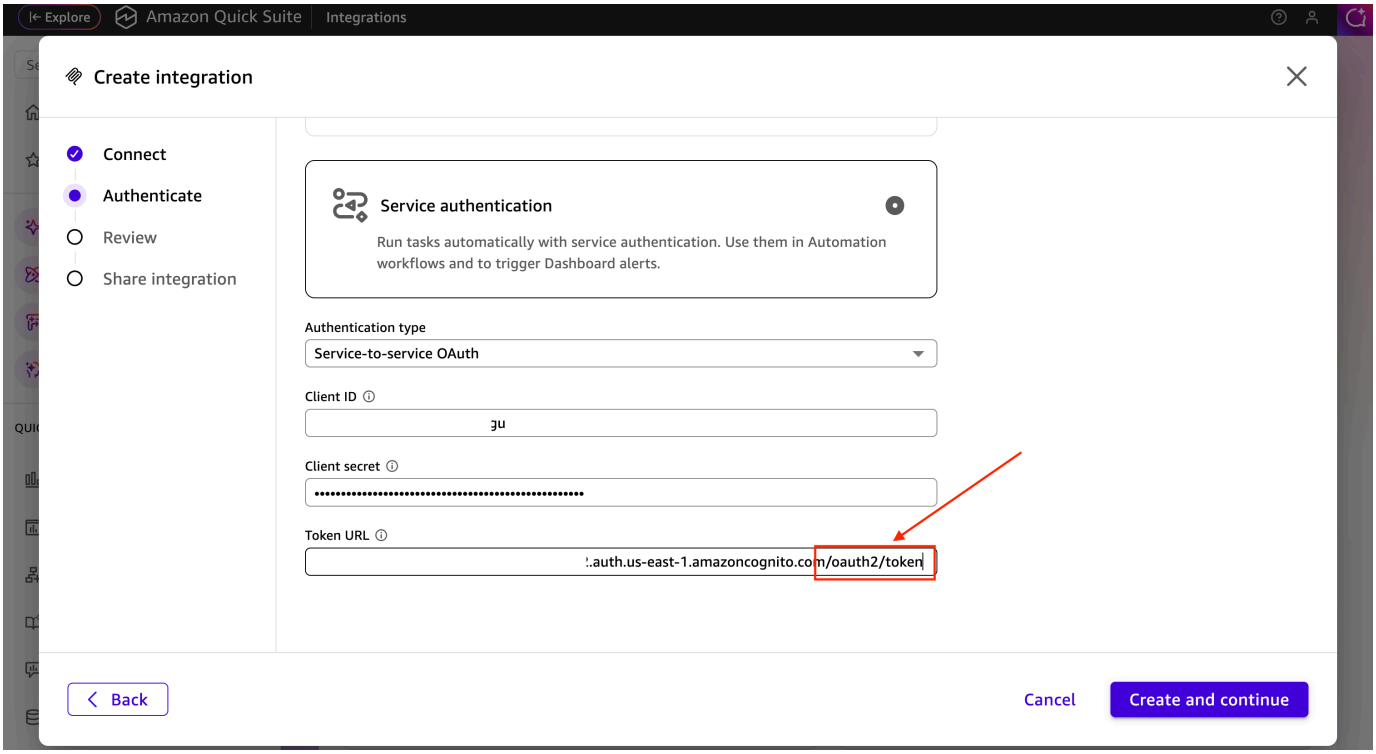
4. Konfigurieren Sie die MCP-Serverdetails:

- MCP-Server-URL: Ihr DLT-MCP-Endpunkt



- Authentifizierungstyp: Dienstbasierte Authentifizierung

- Token-Endpunkt: Ihre Cognito-Token-Endpunkt-URL
- Client-ID: Die Client-ID des M2M-App-Clients
- Geheimer Client-Schlüssel: Der geheime Client-Schlüssel aus dem M2M-App-Client



5. Speichern Sie die MCP-Serveraktionskonfiguration.
6. Fügen Sie Ihrem Agenten die neue MCP-Serveraktion hinzu.

Starten und testen Sie den Agenten

1. Starten Sie den Agenten in Amazon Q Suite.
2. Starten Sie mithilfe von Eingabeaufforderungen in natürlicher Sprache eine Konversation mit dem Agenten.
3. Der Agent verwendet die MCP-Tools, um Ihre Lasttestdaten abzurufen und zu analysieren.

Beispiele für Prompts

Die folgenden Beispiele zeigen, wie Sie mit Ihrem KI-Assistenten interagieren können, um Lasttestdaten über die MCP-Schnittstelle zu analysieren. Passen Sie den Test IDs, die Datumsbereiche und die Kriterien an Ihre spezifischen Testanforderungen an.

Ausführliche Informationen zu den verfügbaren MCP-Tools und ihren Parametern finden Sie in der MCP-Tool-Spezifikation im Developer Guide.

Einfache Abfrage der Testergebnisse

Die Interaktion mit dem MCP-Server in natürlicher Sprache kann so einfach wie `Show me the load tests that have completed in the last 24 hours with their associated completion status` oder aussagekräftiger sein, z. B.

```
Use list_scenarios to find my load tests. Then use get_latest_test_run to show me the
 basic execution data and performance metrics for the most recent test. If the results
 look concerning, also get the detailed performance metrics using get_test_run.
```

Interaktive Leistungsanalyse mit schrittweiser Offenlegung

```
I need to analyze my load test performance, but I'm not sure which specific tests to
 focus on. Please help me by:

1. First, use list_scenarios to show me available test scenarios
2. Ask me which tests I want to analyze based on the list you show me
3. For my selected tests, use list_test_runs to get the test run history
4. Then use get_test_run with the test_run_id to get detailed response times,
 throughput, and error rates
5. If I want to compare tests, use get_baseline_test_run to compare against the
 baseline
6. If there are any issues, use get_test_run_artifacts to help me understand what went
 wrong

Please guide me through this step by step, asking for clarification whenever you need
 more specific information.
```

Validierung der Produktionsreife

```
Help me validate if my API is ready for production deployment:

1. Use list_scenarios to find recent test scenarios
2. For the most recent test scenario, use get_latest_test_run to get basic execution
 data
3. Use get_test_run with that test_run_id to get detailed response times, error rates,
 and throughput
4. Use get_scenario_details with the test_id to show me what load patterns and
 endpoints were tested
```

```
5. If I have a baseline, use get_baseline_test_run to compare current results with the
 baseline
6. Provide a clear go/no-go recommendation based on the performance data
7. If there are any concerns, use get_test_run_artifacts to help identify potential
 issues

My SLA requirements are: response time under [X]ms, error rate under [Y]%.
```

Analyse der Leistungstrends

```
Analyze the performance trend for my load tests over the past [TIME_PERIOD]:

1. Use list_scenarios to get all test scenarios
2. For each scenario, use list_test_runs with start_date and end_date to get tests from
 that period
3. Use get_test_run for the key test runs to get detailed metrics
4. Use get_baseline_test_run to compare against the baseline
5. Identify any significant changes in response times, error rates, or throughput
6. If you detect performance degradation, use get_test_run_artifacts on the problematic
 tests to help identify causes
7. Present the trend analysis in a clear format showing whether performance is
 improving, stable, or degrading

Focus on completed tests and limit results to [N] tests if there are too many.
```

Behebung fehlgeschlagener Tests

```
Help me troubleshoot my failed load tests:

1. Use list_scenarios to find test scenarios
2. For each scenario, use list_test_runs to find recent test runs
3. Use get_test_run with the test_run_id to get the basic execution data and failure
 information
4. Use get_test_run_artifacts to get detailed error messages and logs
5. Use get_scenario_details to understand what was being tested when it failed
6. If I have a similar test that passed, use get_baseline_test_run to identify
 differences
7. Summarize the causes of failure and suggest next steps for resolution

Show me the most recent [N] failed tests from the past [TIME_PERIOD].
```

Entwicklerhandbuch

Dieser Abschnitt enthält den Quellcode für die Lösung und weitere Anpassungen.

Quellcode

Besuchen Sie unser GitHub Repository, um die Vorlagen und Skripte für diese Lösung herunterzuladen und Ihre Anpassungen mit anderen zu teilen.

Wartung

Diese Lösung verwendet Docker-Images mit festen Versionen, die jeder Lösungsversion entsprechen. Standardmäßig sind automatische Updates deaktiviert, sodass Sie die vollständige Kontrolle darüber haben, wann und welche Versionsupdates auf Ihre Bereitstellung angewendet werden. Das AWS-Lösungsteam verwendet Amazon ECR Enhanced Scanning, um häufig auftretende Sicherheitslücken und Risiken (CVEs) im Basis-Image und in den installierten Paketen zu erkennen. Wenn möglich, veröffentlicht das Team gepatchte Images mit demselben Versions-Tag, um das Problem zu beheben, CVEs ohne die Kompatibilität mit der veröffentlichten Lösungsversion zu beeinträchtigen.

Wenn Bilder auf derselben Nebenversion gepatcht werden, wird das Stable-Tag automatisch aktualisiert, und ein zusätzliches Image-Tag wird in diesem Format erstellt. `<solution-version>_<date-of-fix>` Wenn eine Haupt- oder Nebenversion veröffentlicht wird, müssen Sie ein Full-Stack-Update durchführen, um die neueste Image-Version zu erhalten, da das Stable-Tag entsprechend der Lösungsversion inkrementiert wird.

Wenn Sie sich während der Bereitstellung für automatische Updates entscheiden, werden Änderungen am Image, einschließlich CVE-Patches und kleinerer Bugfixes, automatisch bis zur neuesten passenden Nebenversion übernommen.

Versionen

Standardmäßig wird diese Lösung mit deaktivierten automatischen Updates bereitgestellt. Das bedeutet, dass die Container-Image-Version an die spezifische Version gebunden ist, die Ihrer bereitgestellten Lösungsversion entspricht, sodass Sie die volle Kontrolle über Versionsupdates haben.

Wenn Sie sich dafür entscheiden, automatische Updates zu aktivieren, indem Sie während der CloudFormation Bereitstellung Ja auswählen, erhält die Lösung automatisch Sicherheitspatches und kleinere, unveränderliche Bugfixes bis zur neuesten passenden Nebenversion. Wenn Sie beispielsweise Version 4.0.0 mit aktivierten automatischen Updates bereitstellen, erhalten Sie Updates bis 4.0.x, aber nicht 4.1.0 oder höher.

Um die Container-Image-Version manuell zu steuern, können Sie die Aufgabendefinition bearbeiten, um eine bestimmte Image-Version im Tag-Versionsformat anzugeben. Auf diese Weise können Sie unabhängig von der Einstellung für automatische Updates an eine bestimmte Image-Version anheften.

Anpassung des Container-Images

Diese Lösung verwendet ein öffentliches Amazon Elastic Container Registry (Amazon ECR) -Image-Repository, das von AWS verwaltet wird, um das Image zu speichern, das für die Ausführung der konfigurierten Tests verwendet wird. Wenn Sie das Container-Image anpassen möchten, können Sie das Image neu erstellen und in ein ECR-Image-Repository in Ihrem eigenen AWS-Konto übertragen.

Wenn Sie diese Lösung anpassen möchten, können Sie das Standard-Container-Image verwenden oder diesen Container nach Ihren Bedürfnissen bearbeiten. Wenn Sie die Lösung anpassen, verwenden Sie das folgende Codebeispiel, um die Umgebungsvariablen zu deklarieren, bevor Sie Ihre benutzerdefinierte Lösung erstellen.

```
#!/bin/bash
export REGION=aws-region-code # the AWS region to launch the solution (e.g. us-east-1)
export BUCKET_PREFIX=my-bucket-name # prefix of the bucket name without the region code
export BUCKET_NAME=$BUCKET_PREFIX-$REGION # full bucket name where the code will reside
export SOLUTION_NAME=my-solution-name
export VERSION=my-version # version number for the customized code
export PUBLIC_ECR_REGISTRY=public.ecr.aws/awssolutions/distributed-load-testing-on-
aws-load-tester # replace with the container registry and image if you want to use a
 different container image export PUBLIC_ECR_TAG=v3.1.0 # replace with the container
 image tag if you want to use a different container image
```

Wenn Sie das Container-Image anpassen möchten, können Sie es entweder in einem privaten Image-Repository oder in einem öffentlichen Image-Repository in Ihrem AWS-Konto hosten. Die Image-Ressourcen befinden sich im `deployment/ecr/distributed-load-testing-on-aws-load-tester` Verzeichnis, das sich in der Codebasis befindet.

Sie können das Image erstellen und an das Host-Ziel übertragen.

- Informationen zu privaten Amazon ECR-Repositorys und -Images finden Sie unter Private Amazon ECR-Repositorys und private Images im Amazon ECR-Benutzerhandbuch.
- Informationen zu öffentlichen Amazon ECR-Repositorys und -Images finden Sie unter Öffentliche Amazon ECR-Repositorys und öffentliche Images im Amazon ECR Public User Guide.

Sobald Sie Ihr eigenes Image erstellt haben, können Sie die folgenden Umgebungsvariablen deklarieren, bevor Sie Ihre maßgeschneiderte Lösung erstellen.

```
#!/bin/bash
export PUBLIC_ECR_REGISTRY=YOUR_ECR_REGISTRY_URI # e.g. YOUR_ACCOUNT_ID.dkr.ecr.us-
east-1.amazonaws.com/YOUR_IMAGE_NAME
export PUBLIC_ECR_TAG=YOUR_ECR_TAG # e.g. latest, v3.4.0
```

Das folgende Beispiel zeigt die Containerdatei.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal

RUN dnf update -y && \
    dnf install -y python3.11 python3.11-pip java-21-amazon-corretto bc procps jq
 findutils unzip && \
    dnf clean all

ENV PIP_INSTALL="pip3.11 install --no-cache-dir"


# install bzt
RUN $PIP_INSTALL --upgrade bzt awscli setuptools==78.1.1 h11 urllib3==2.2.2 && \
    $PIP_INSTALL --upgrade bzt
COPY ./.bzt-rc /root/.bzt-rc
RUN chmod 755 /root/.bzt-rc

# install bzt tools
RUN bzt -install-tools -o modules.install-
checker.exclude=selenium,gatling,tsung,siege,ab,k6,external-results-
loader,locust,junit,testng,rspec,mocha,nunit,xunit,wdio,robot,newman
RUN rm -rf /root/.bzt/selenium-taurus
RUN mkdir /bzt-configs /tmp/artifacts
ADD ./load-test.sh /bzt-configs/
ADD ./*.jar /bzt-configs/
ADD ./*.py /bzt-configs/
```

```
RUN chmod 755 /bzt-configs/load-test.sh
RUN chmod 755 /bzt-configs/ecslistener.py
RUN chmod 755 /bzt-configs/ecscontroller.py
RUN chmod 755 /bzt-configs/jar_updater.py
RUN python3.11 /bzt-configs/jar_updater.py

# Remove jar files from /tmp
RUN rm -rf /tmp/jmeter-plugins-manager-1* && \
    rm -rf /usr/local/lib/python3.11/site-packages/setuptools-65.5.0.dist-info && \
    rm -rf /usr/local/lib/python3.11/site-packages/urllib3-1.26.17.dist-info

# Add settings file to capture the output logs from bzt cli
RUN mkdir -p /etc/bzt.d && echo '{"settings": {"artifacts-dir": "/tmp/artifacts"}}' > /
etc/bzt.d/90-artifacts-dir.json

WORKDIR /bzt-configs
ENTRYPOINT ["./load-test.sh"]
```

Zusätzlich zu einer Container-Datei enthält das Verzeichnis das folgende Bash-Skript, das die Testkonfiguration von Amazon S3 herunterlädt, bevor das Taurus/Blazemeter Programm ausgeführt wird.

```
#!/bin/bash

# set a uuid for the results xml file name in S3
UUID=$(cat /proc/sys/kernel/random/uuid)
pypid=0
echo "S3_BUCKET:: ${S3_BUCKET}"
echo "TEST_ID:: ${TEST_ID}"
echo "TEST_TYPE:: ${TEST_TYPE}"
echo "FILE_TYPE:: ${FILE_TYPE}"
echo "PREFIX:: ${PREFIX}"
echo "UUID:: ${UUID}"
echo "LIVE_DATA_ENABLED:: ${LIVE_DATA_ENABLED}"
echo "MAIN_STACK_REGION:: ${MAIN_STACK_REGION}"

cat /proc/self/cgroup
TASK_ID=$(grep -oE '[a-f0-9]{32}' /proc/self/cgroup | head -n 1)
echo $TASK_ID

sigterm_handler() {
  if [ $pypid -ne 0 ]; then
    echo "container received SIGTERM."
```

```
    kill -15 $pypid
    wait $pypid
    exit 143 #128 + 15
  fi
}
trap 'sigterm_handler' SIGTERM

echo "Download test scenario"
aws s3 cp s3://$S3_BUCKET/test-scenarios/$TEST_ID-$AWS_REGION.json test.json --region
 $MAIN_STACK_REGION

# Set the default log file values to jmeter
LOG_FILE="jmeter.log"
OUT_FILE="jmeter.out"
ERR_FILE="jmeter.err"
KPI_EXT="jtl"

# download JMeter jmx file
if [ "$TEST_TYPE" != "simple" ]; then
  # setting the log file values to the test type
  LOG_FILE="${TEST_TYPE}.log"
  OUT_FILE="${TEST_TYPE}.out"
  ERR_FILE="${TEST_TYPE}.err"

  # set variables based on TEST_TYPE
  if [ "$TEST_TYPE" == "jmeter" ]; then
    EXT="jmx"
    TYPE_NAME="JMeter"
    # Copy *.jar to JMeter library path. See the Taurus JMeter path: https://
gettaurus.org/docs/JMeter/
    JMETER_LIB_PATH=`find ~/.bzt/jmeter-taurus -type d -name "lib"`
    echo "cp $PWD/*.jar $JMETER_LIB_PATH"
    cp $PWD/*.jar $JMETER_LIB_PATH
  elif [ "$TEST_TYPE" == "k6" ]; then
    curl --output /tmp/artifacts/k6.rpm https://dl.k6.io/rpm/x86_64/k6-v0.58.0-
amd64.rpm
    rpm -ivh /tmp/artifacts/k6.rpm
    dnf install -y k6
    rm -rf /tmp/artifacts/k6.rpm
    EXT="js"
    KPI_EXT="csv"
    TYPE_NAME="K6"
  elif [ "$TEST_TYPE" == "locust" ]; then
    EXT="py"
```

```
    TYPE_NAME="Locust"

  fi

  if [ "$FILE_TYPE" != "zip" ]; then
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.$EXT ./ --
region $MAIN_STACK_REGION
  else
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.zip ./ --region
 $MAIN_STACK_REGION
    unzip $TEST_ID.zip
    echo "UNZIPPED"
    ls -l

    # If zip and locust, make sure to pick locustfile
    if [ "$TEST_TYPE" != "locust" ]; then
      TEST_SCRIPT=$(find . -name "*.${EXT}" | head -n 1)
    else
      TEST_SCRIPT=$(find . -name "locustfile.py" | head -n 1)
    fi
    # only looks for the first test script file.
    TEST_SCRIPT=`find . -name "*.${EXT}" | head -n 1`
    echo $TEST_SCRIPT
    if [ -z "$TEST_SCRIPT" ]; then
      echo "There is no test script (.${EXT}) in the zip file."
      exit 1
    fi

    sed -i -e "s|$TEST_ID.$EXT|$TEST_SCRIPT|g" test.json

    # copy bundled plugin jars to jmeter extension folder to make them available to
 jmeter
    BUNDLED_PLUGIN_DIR=`find $PWD -type d -name "plugins" | head -n 1`
    # attempt to copy only if a /plugins folder is present in upload
    if [ -z "$BUNDLED_PLUGIN_DIR" ]; then
      echo "skipping plugin installation (no /plugins folder in upload)"
    else
      # ensure the jmeter extensions folder exists
      JMETER_EXT_PATH=`find ~/.bzt/jmeter-taurus -type d -name "ext"`
      if [ -z "$JMETER_EXT_PATH" ]; then
        # fail fast - if plugins bundled they will be needed for the tests
        echo "jmeter extension path (~/.bzt/jmeter-taurus/**/ext) not found - cannot
 install bundled plugins"
        exit 1
```

```
      fi
      cp -v $BUNDLED_PLUGIN_DIR/*.jar $JMETER_EXT_PATH
    fi
  fi
fi

#Download python script
if [ -z "$IPNETWORK" ]; then
    python3.11 -u $SCRIPT  $TIMEOUT &
    pypid=$!
    wait $pypid
    pypid=0
else
    aws s3 cp s3://$S3_BUCKET/Container_IPs/${TEST_ID}_IPHOSTS_${AWS_REGION}.txt ./ --
region $MAIN_STACK_REGION
    export IPHOSTS=$(cat ${TEST_ID}_IPHOSTS_${AWS_REGION}.txt)
    python3.11 -u $SCRIPT $IPNETWORK $IPHOSTS
fi

echo "Running test"

stdbuf -i0 -o0 -e0 bzt test.json -o modules.console.disable=true | stdbuf -i0 -o0 -e0
 tee -a result.tmp | sed -u -e "s|^|$TEST_ID $LIVE_DATA_ENABLED |"
CALCULATED_DURATION=`cat result.tmp | grep -m1 "Test duration" | awk -F ' ' '{ print
 $5 }' | awk -F ':' '{ print ($1 * 3600) + ($2 * 60) + $3 }'`

# upload custom results to S3 if any
# every file goes under $TEST_ID/$PREFIX/$UUID to distinguish the result correctly
if [ "$TEST_TYPE" != "simple" ]; then
  if [ "$FILE_TYPE" != "zip" ]; then
    cat $TEST_ID.$EXT | grep filename > results.txt
  else
    cat $TEST_SCRIPT | grep filename > results.txt
  fi

  if [ -f results.txt ]; then
    sed -i -e 's/<stringProp name="filename">//g' results.txt
    sed -i -e 's/<\/stringProp>//g' results.txt
    sed -i -e 's/ //g' results.txt

    echo "Files to upload as results"
    cat results.txt

    files=(`cat results.txt`)
```

```
    extensions=()
    for f in "${files[@]}"; do
      ext="${f##*.}"
      if [[ ! " ${extensions[@]} " =~ " ${ext} " ]]; then
        extensions+=("$ext")
      fi
    done

    # Find all files in the current folder with the same extensions
    all_files=()
    for ext in "${extensions[@]}"; do
      for f in *."$ext"; do
        all_files+=("$f")
      done
    done

    for f in "${all_files[@]}"; do
      p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID/$f"
      if [[ $f = /* ]]; then
        p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID$f"
      fi

        echo "Uploading $p"
        aws s3 cp $f $p --region $MAIN_STACK_REGION
    done
    fi
fi

if [ -f /tmp/artifacts/results.xml ]; then

  # Insert the Task ID at the same level as <FinalStatus>
  curl -s $ECS_CONTAINER_METADATA_URI_V4/task
  Task_CPU=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.CPU')
  Task_Memory=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.Memory')
  START_TIME=$(curl -s "$ECS_CONTAINER_METADATA_URI_V4/task" | jq -r
 '.Containers[0].StartedAt')
  # Convert start time to seconds since epoch
  START_TIME_EPOCH=$(date -d "$START_TIME" +%s)
  # Calculate elapsed time in seconds
  CURRENT_TIME_EPOCH=$(date +%s)
  ECS_DURATION=$((CURRENT_TIME_EPOCH - START_TIME_EPOCH))
```

```
  sed -i.bak 's/<\/FinalStatus>/<TaskId>'"$TASK_ID"'<\/TaskId><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskCPU>'"$Task_CPU"'<\/TaskCPU><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskMemory>'"$Task_Memory"'<\/TaskMemory><\/
FinalStatus>/' /tmp/artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<ECSDuration>'"$ECS_DURATION"'<\/ECSDuration><\/
FinalStatus>/' /tmp/artifacts/results.xml

  echo "Validating Test Duration"
  TEST_DURATION=$(grep -E '<TestDuration>[0-9]+.[0-9]+</TestDuration>' /tmp/artifacts/
results.xml | sed -e 's/<TestDuration>//' | sed -e 's/<\/TestDuration>//')

  if (( $(echo "$TEST_DURATION > $CALCULATED_DURATION" | bc -l) )); then
    echo "Updating test duration: $CALCULATED_DURATION s"
    sed -i.bak.td 's/<TestDuration>[0-9]*\.[0-9]*<\/TestDuration>/
<TestDuration>'"$CALCULATED_DURATION"'<\/TestDuration>/' /tmp/artifacts/results.xml
  fi

  if [ "$TEST_TYPE" == "simple" ]; then
    TEST_TYPE="jmeter"
  fi

  echo "Uploading results, bzt log, and JMeter log, out, and err files"
  aws s3 cp /tmp/artifacts/results.xml s3://$S3_BUCKET/results/${TEST_ID}/${PREFIX}-
${UUID}-${AWS_REGION}.xml --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/bzt.log s3://$S3_BUCKET/results/${TEST_ID}/bzt-${PREFIX}-
${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$LOG_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$OUT_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.out --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$ERR_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.err --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/kpi.${KPI_EXT} s3://$S3_BUCKET/results/${TEST_ID}/kpi-
${PREFIX}-${UUID}-${AWS_REGION}.${KPI_EXT} --region $MAIN_STACK_REGION

else
  echo "An error occurred while the test was running."
fi
```

Neben dem Dockerfile und dem Bash-Skript sind auch zwei Python-Skripte im Verzeichnis enthalten. Jede Aufgabe führt ein Python-Skript innerhalb des Bash-Skripts aus. Die Worker-Tasks führen das

`ecslistener.py` Skript aus, während die Leader-Task das `ecscontroller.py` Skript ausführt. Das `ecslistener.py` Skript erstellt einen Socket auf Port 50000 und wartet auf eine Nachricht. Das `ecscontroller.py` Skript stellt eine Verbindung zum Socket her und sendet die Starttestnachricht an die Worker-Aufgaben, sodass sie gleichzeitig starten können.

API zum Testen verteilter Lasten

Diese Lasttestlösung hilft Ihnen dabei, Testergebnisdaten auf sichere Weise verfügbar zu machen. Die API fungiert als „Eingangstür“ für den Zugriff auf Testdaten, die in Amazon DynamoDB gespeichert sind. Sie können die auch verwenden APIs , um auf alle erweiterten Funktionen zuzugreifen, die Sie in die Lösung integriert haben.

Diese Lösung verwendet einen Amazon Cognito Cognito-Benutzerpool, der in Amazon API Gateway zur Identifizierung und Autorisierung integriert ist. Wenn ein Benutzerpool mit der API verwendet wird, dürfen Clients nur Methoden aufrufen, die vom Benutzerpool aktiviert wurden, nachdem sie ein gültiges Identitätstoken bereitgestellt haben.

Weitere Informationen zur Ausführung von Tests direkt über die API finden Sie unter Signing Requests in der Amazon API Gateway REST API-Referenzdokumentation.

Die folgenden Operationen sind in der API der Lösung verfügbar.

> **Note**
>
> Weitere Informationen zu `testScenario` und anderen Parametern finden Sie in den Szenarien und Payload-Beispielen im GitHub Repository.

Informationen zum Stapel

- ERHALTE /stack-info

Szenarien

- GET /scenarios
- POST /szenarien
- OPTIONEN /Szenarien
- GET /scenarios/ {testId}

- POSTE /scenarios/ {testId}
- LÖSCHEN SIE /scenarios/ {testId}
- OPTIONEN /scenarios/ {testID}

Testläufe

- GET /scenarios/ {testId} /testruns
- GET /scenarios/ {testID} /testruns/ {} testRunId
- LÖSCHE /scenarios/ {testId} /testruns/ {testRunId}

Basislinie

- GET /scenarios/ {testId} /baseline
- GIB /scenarios/ {testId} /baseline ein
- LÖSCHEN SIE /scenarios/ {testId} /baseline

Aufgaben

- HOLEN SIE SICH /tasks
- OPTIONEN /Aufgaben

Regionen

- HOLEN SIE SICH /regions
- OPTIONEN /regions

ERHALTE /stack-info

Description

Der `GET /stack-info` Vorgang ruft Informationen über den bereitgestellten Stack ab, einschließlich Erstellungszeit, Region und Version. Dieser Endpunkt wird vom Frontend verwendet.

Antwort

200 — Erfolg

Name	Description
`created_time`	ISO 8601-Zeitstempel, als der Stack erstellt wurde (zum Beispiel) `2025-09-09T19:40:22Z`
`region`	AWS-Region, in der der Stack bereitgestellt wird (z. B.`us-east-1`)
`version`	Version der bereitgestellten Lösung (z. B.`v4.0.0`)

Antworten auf Fehler

- `403`- Verboten: Unzureichende Berechtigungen für den Zugriff auf Stack-Informationen
- `404`- Nicht gefunden: Stack-Informationen nicht verfügbar
- `500`- Interner Serverfehler

GET /scenarios

Description

Die `GET /scenarios` Operation ermöglicht es Ihnen, eine Liste von Testszenarien abzurufen.

Antwort

Name	Description
`data`	Eine Liste von Szenarien, einschließlich der ID, des Namens, der Beschreibung, des Status, der Laufzeit, der Tags, der Gesamtzahl der Testläufe und der letzten Ausführung für jeden Test

POST /szenarien

Description

Die `POST /scenarios` Operation ermöglicht es Ihnen, ein Testszenario zu erstellen oder zu planen.

Anforderungstext

Name	Description
`testName`	Der Name des Tests
`testDescription`	Die Beschreibung des Tests
`testTaskConfigs`	Ein Objekt, das `concurrency` (die Anzahl der parallel Läufe), `taskCount` (die Anzahl der Aufgaben, die zur Ausführung eines Tests erforderlich sind) und `region` für das Szenario angibt
`testScenario`	Die Testspezifikation, einschließlich Parallelität, Testzeit, Host und Methode für den Test
`testType`	Der Testtyp (zum Beispiel`simple`,`jmeter`)
`fileType`	Der Upload-Dateityp (z. B.`none`,`script`,`zip`)
`tags`	Eine Reihe von Zeichenketten zur Kategorisierung von Tests. Optionales Feld mit einer maximalen Länge von 5 (z. B.`["blue", "3.0", "critical"]`)
`scheduleDate`	Das Datum, an dem ein Test ausgeführt werden soll. Wird nur angegeben, wenn ein Test geplant wird (z. B.`2021-02-28`)
`scheduleTime`	Die Zeit, um einen Test durchzuführen. Wird nur angegeben, wenn ein Test geplant wird `21:07` (z. B.

Name	Description
`scheduleStep`	Der Schritt im Planungsprozess. Wird nur bereitgestellt, wenn ein wiederkehrender Test geplant wird. (Zu den verfügbaren Schritten gehören `create` und`start`)
`cronvalue`	Der Cron-Wert für die Anpassung der wiederkehrenden Terminplanung. Falls verwendet, lassen Sie ScheduleDate und ScheduleTime weg.
`cronExpiryDate`	Erforderliches Datum, damit der Cron abläuft und nicht unbegrenzt läuft.
`recurrence`	Die Wiederholung eines geplanten Tests. Wird nur bereitgestellt, wenn ein wiederkeh render Test geplant wird (z. B.`daily`, `weekly`, `biweekly`, oder`monthly`)

Antwort

Name	Description
`testId`	Die eindeutige ID des Tests
`testName`	Der Name des Tests
`status`	Der Status des Tests

OPTIONEN/Szenarien

Description

Die `OPTIONS /scenarios` Operation liefert eine Antwort auf die Anfrage mit den richtigen CORS-Antwortheadern.

Antwort

Name	Description
`testId`	Die eindeutige ID des Tests
`testName`	Der Name des Tests
`status`	Der Status des Tests

GET /scenarios/ {testId}

Description

Die `GET /scenarios/{testId}` Operation ermöglicht es Ihnen, die Details eines bestimmten Testszenarios abzurufen.

Anforderungsparameter

`testId`

- Die eindeutige ID des Tests

  Typ: Zeichenfolge

  Erforderlich: Ja

`latest`

- Abfrageparameter, um nur den letzten Testlauf zurückzugeben. Die Standardeinstellung ist `true`

  Typ: Boolesch

  Erforderlich: Nein

`history`

- Abfrageparameter, um den Testlaufverlauf in die Antwort aufzunehmen. Der Standardwert ist `true`. Auf einstellen, `false` um den Verlauf auszuschließen

  Typ: Boolesch

Erforderlich: Nein

Antwort

Name	Description
`testId`	Die eindeutige ID des Tests
`testName`	Der Name des Tests
`testDescription`	Die Beschreibung des Tests
`testType`	Die Art des Tests, der ausgeführt wird (z. B.`simple`,`jmeter`)
`fileType`	Der Typ der Datei, die hochgeladen wird (z. B.`none`,`script`,`zip`)
`tags`	Eine Reihe von Zeichenketten zur Kategorisierung von Tests
`status`	Der Status des Tests
`startTime`	Die Uhrzeit und das Datum, an dem der letzte Test gestartet wurde
`endTime`	Uhrzeit und Datum, an dem der letzte Test beendet wurde
`testScenario`	Die Testspezifikation, einschließlich Parallelität, Testzeit, Host und Methode für den Test
`taskCount`	Die Anzahl der Aufgaben, die zur Ausführung des Tests erforderlich sind
`taskIds`	Eine Liste von Aufgaben IDs zum Ausführen von Tests
`results`	Die endgültigen Ergebnisse des Tests

Name	Description
`history`	Eine Liste der Endergebnisse früherer Tests (ausgenommen wann`history=false`)
`totalRuns`	Die Gesamtzahl der Testläufe für dieses Szenario
`lastRun`	Der Zeitstempel des letzten Testlaufs
`errorReason`	Eine Fehlermeldung, die generiert wird, wenn ein Fehler auftritt
`nextRun`	Der nächste geplante Lauf (zum Beispiel`2017-04-22 17:18:00`)
`scheduleRecurrence`	Die Wiederholung des Tests (z. B.,`daily`, `weeklybiweekly`,`monthly`)

POST /scenarios/ {TestID}

Description

Der `POST /scenarios/{testId}` Vorgang ermöglicht es Ihnen, ein bestimmtes Testszenario abzubrechen.

Parameter anfordern

`testId`

- Die eindeutige ID des Tests

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

Name	Description
`status`	Der Status des Tests

LÖSCHEN SIE /scenarios/ {testId}

Description

Die `DELETE /scenarios/{testId}` Operation ermöglicht es Ihnen, alle Daten zu löschen, die sich auf ein bestimmtes Testszenario beziehen.

Parameter anfordern

`testId`

- Die eindeutige ID des Tests

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

Name	Description
`status`	Der Status des Tests

OPTIONEN /scenarios/ {testId}

Description

Die `OPTIONS /scenarios/{testId}` Operation liefert eine Antwort auf die Anfrage mit den richtigen CORS-Antwortheadern.

Antwort

Name	Description
`testId`	Die eindeutige ID des Tests
`testName`	Der Name des Tests
`testDescription`	Die Beschreibung des Tests
`testType`	Die Art des Tests, der ausgeführt wird (z. B.`simple`,`jmeter`)
`fileType`	Der Typ der Datei, die hochgeladen wird (z. B.`none`,`script`,`zip`)
`status`	Der Status des Tests
`startTime`	Die Uhrzeit und das Datum, an dem der letzte Test gestartet wurde
`endTime`	Uhrzeit und Datum, an dem der letzte Test beendet wurde
`testScenario`	Die Testspezifikation, einschließlich Parallelität, Testzeit, Host und Methode für den Test
`taskCount`	Die Anzahl der Aufgaben, die zur Ausführung des Tests erforderlich sind
`taskIds`	Eine Liste von Aufgaben IDs zum Ausführen von Tests
`results`	Die endgültigen Ergebnisse des Tests
`history`	Eine Liste der Endergebnisse vergangener Tests
`errorReason`	Eine Fehlermeldung, die generiert wird, wenn ein Fehler auftritt

GET /scenarios/ {testId} /testruns

Description

Der `GET /scenarios/{testId}/testruns` Vorgang ruft einen Testlauf IDs für ein bestimmtes Testszenario ab, optional gefiltert nach Zeitbereich. Wann`latest=true`, gibt nur den letzten Testlauf zurück.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

`latest`

- Gibt nur die letzte Testlauf-ID zurück

  Typ: Boolescher Wert

  Standard: `false`

  Erforderlich: Nein

`start_timestamp`

- ISO 8601-Zeitstempel zum Filtern von Testläufen (einschließlich). Beispiel: `2024-01-01T00:00:00Z`

  Typ: Zeichenfolge (Datums-/Uhrzeitformat)

  Erforderlich: Nein

`end_timestamp`

- ISO 8601-Zeitstempel zum Filtern von Testläufen bis (einschließlich). Beispiel: `2024-12-31T23:59:59Z`

  Typ: Zeichenfolge (Datums-/Uhrzeitformat)

  Erforderlich: Nein

`limit`

- Maximale Anzahl zurückzugebender Testläufe (wird ignoriert, wenn) `latest=true`

  Typ: Integer (mindestens: 1, maximal: 100)

  Standard: `20`

  Erforderlich: Nein

`next_token`

- Paginierungstoken aus der vorherigen Antwort, um die nächste Seite zu erhalten

  Typ: Zeichenfolge

  Erforderlich: Nein

Antwort

200 — Erfolg

Name	Description
`testRuns`	Array von Testlaufobjekten, die jeweils `testRunId` (Zeichenfolge) und `startTime` (ISO 8601 Datum/Uhrzeit) enthalten
`pagination`	Objekt, das `limit` (Ganzzahl) und `next_token` (Zeichenfolge oder Null) enthält. Das Token ist Null, wenn keine weiteren Ergebnisse vorliegen

Antworten auf Fehler

- `400`- Ungültiges Zeitstempelformat oder ungültige Parameter
- `404`- Das Testszenario wurde nicht gefunden
- `500`- Interner Serverfehler

Beispielverwendung

- Nur letzter Testlauf: `GET /scenarios/test123/testruns?latest=true`
- Spätester innerhalb des Zeitbereichs: `GET /scenarios/test123/testruns?latest=true&start_timestamp=2024-01-01T00:00:00Z`
- Anfrage zur nächsten Seite: `GET /scenarios/test123/testruns?limit=20&next_token=eyJ0ZXN0SWQiOiJzZVFVeTEyTEtMIiwic3RhcnRUaW1lIjoiMjAyNC0wMS`

GET /scenarios/ {testId} /testruns/ {} testRunId

Description

Der `GET /scenarios/{testId}/testruns/{testRunId}` Vorgang ruft vollständige Ergebnisse und Metriken für einen bestimmten Testlauf ab. Lassen Sie optional Verlaufsergebnisse aus, um schneller reagieren `history=false` zu können.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

`testRunId`

- Die spezifische Testlauf-ID

  Typ: Zeichenfolge

  Erforderlich: Ja

`history`

- Schließt das Verlaufs-Array als Antwort ein. Stellen Sie auf ein`false`, um den Verlauf für eine schnellere Antwort auszulassen

  Typ: Boolescher Wert

  Standard: `true`

Erforderlich: Nein

Antwort

200 — Erfolgreich

Name	Description
`testId`	Die eindeutige ID des Tests (zum Beispiel`seQUy12LKL`)
`testRunId`	Die spezifische Testlauf-ID (z. B.`2DEwHItEn e`)
`testDescription`	Beschreibung des Belastungstests
`testType`	Die Art des Tests (z. B.`simple`,`jmeter`)
`status`	Der Status des Testlaufs: `completerunning`,`failed`, oder `cancelled`
`startTime`	Die Uhrzeit und das Datum, an dem der Test gestartet wurde (z. B.`2025-09-09 21:01:00`)
`endTime`	Uhrzeit und Datum, an dem der Test beendet wurde (z. B.`2025-09-09 21:18:29`)
`succPercent`	Erfolgsquote (zum Beispiel`100.00`)
`testTaskConfigs`	Array von Aufgabenkonfigurationsobjek tenregion, die`taskCount` , und enthalten `concurrency`
`completeTasks`	Objekte ordnen Regionen der Anzahl abgeschlossener Aufgaben zu

Name	Description
`results`	Objekt mit detaillierten Kennzahlen wie `avg_lt` (durchschnittliche Latenz), Perzentile (`p0_0`,,,,`p50_0`,`p90_0`,`p100_0`) `p95_0` `p99_0``p99_9`, `avg_rt` (durchschnittliche Reaktionszeit), (durchschnittliche Verbindungszeit), `avg_ct` (durchschnittliche Verbindungszeit), `stdev_rt` (Reaktionszeit mit Standardabweichung),, `concurrency throughput` , `succ` (Anzahl der Erfolge), `fail` (Anzahl der Fehler),,, `bytes` `testDuration` `metricS3Location` , `rc` (Antwortcode-Array) und Array `labels`
`testScenario`	Objekt, das eine Testkonfiguration mit Eigenschaften`execution` , und `reporting` enthält `scenarios`
`history`	Array mit historischen Testergebnissen (ausgenommen wann`history=false`)

Antworten auf Fehler

- `400`- Ungültige TestID oder testRunId
- `404`- Testlauf nicht gefunden
- `500`- Interner Serverfehler

LÖSCHEN Sie /scenarios/ {testId} /testruns/ {} testRunId

Description

Der `DELETE /scenarios/{testId}/testruns/{testRunId}` Vorgang löscht alle Daten und Artefakte, die sich auf einen bestimmten Testlauf beziehen. Die Testlaufdaten werden aus DynamoDB entfernt, während die tatsächlichen Testdaten in S3 unverändert bleiben.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

`testRunId`

- Die spezifische Testlauf-ID, die gelöscht werden soll

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

204 — Erfolgreich

Testlauf wurde erfolgreich gelöscht (es wurde kein Inhalt zurückgegeben)

Antworten auf Fehler

- `400`- Ungültige TestID oder testRunId
- `403`- Verboten: Unzureichende Berechtigungen zum Löschen des Testlaufs
- `404`- Testlauf nicht gefunden
- `409`- Konflikt: Der Testlauf läuft gerade und kann nicht gelöscht werden
- `500`- Interner Serverfehler

GET /scenarios/ {testId} /baseline

Description

Der `GET /scenarios/{testId}/baseline` Vorgang ruft das angegebene Baseline-Testergebnis für ein Szenario ab. Gibt je nach `data` Parameter entweder die ID des Ausgangstestlaufs oder die vollständigen Baseline-Ergebnisse zurück.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

`data`

- Gibt die vollständigen Basisdaten des Testlaufs zurück`true`, wenn, andernfalls nur testRunId

  Typ: Boolescher Wert

  Standard: `false`

  Erforderlich: Nein

Antwort

200 — Erfolgreich

Wann `data=false` (Standard):

Name	Description
`testId`	Die ID des Testszenarios (zum Beispiel`eQUy12LKL`)
`baselineTestRunId`	Die ID des Ausgangstestlaufs (z. B.`2DEwHItEne`)

Wann`data=true`:

Name	Description
`testId`	Die ID des Testszenarios (zum Beispiel`eQUy12LKL`)

Name	Description
`baselineTestRunId`	Die ID des Ausgangstestlaufs (z. B.`2DEwHItEne`)
`baselineData`	Vollständiges Objekt mit den Ergebnissen des Testlaufs (gleiche Struktur wie`GET /scenarios/{testId}/testruns/{testRunId}`)

Antworten auf Fehler

- `400`- Ungültiger TestID-Parameter
- `404`- Das Testszenario wurde nicht gefunden oder es wurde kein Basiswert festgelegt
- `500`- Interner Serverfehler

PUT /scenarios/ {TestId} /baseline

Description

Der `PUT /scenarios/{testId}/baseline` Vorgang legt einen bestimmten Testlauf als Grundlage für den Leistungsvergleich fest. Pro Szenario kann nur eine Basislinie festgelegt werden.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

Anforderungstext

Name	Description
`testRunId`	Die Testlauf-ID, die als Ausgangswert festgeleg t werden soll (z. B.`2DEwHItEne`)

Antwort

200 — Erfolgreich

Name	Description
`message`	Bestätigungsnachricht (zum Beispiel`Baseline set successfully`)
`testId`	Die ID des Testszenarios (zum Beispiel`seQUy12LKL`)
`baselineTestRunId`	Die eingestellte Baseline-Testlauf-ID (z. B.`2DEwHItEne`)

Antworten auf Fehler

- `400`- Ungültige TestID oder testRunId
- `404`- Testszenario oder Testlauf nicht gefunden
- `409`- Konflikt: Der Testlauf kann nicht als Ausgangswert festgelegt werden (z. B. fehlgeschlagener Test)
- `500`- Interner Serverfehler

LÖSCHEN Sie /scenarios/ {testId} /baseline

Description

Die `DELETE /scenarios/{testId}/baseline` Operation löscht den Basiswert für ein Szenario, indem er auf eine leere Zeichenfolge gesetzt wird.

Anforderungsparameter

`testId`

- Die ID des Testszenarios

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

204 — Erfolg

Baseline wurde erfolgreich gelöscht (es wurde kein Inhalt zurückgegeben)

Antworten auf Fehler

- `400`- Ungültige TestID
- `500`- Interner Serverfehler

GET /tasks

Description

Mit `GET /tasks` diesem Vorgang können Sie eine Liste der laufenden Amazon Elastic Container Service (Amazon ECS) -Aufgaben abrufen.

Antwort

Name	Description
`tasks`	Eine Liste von Aufgaben IDs zum Ausführen von Tests

OPTIONEN/Aufgaben

Description

Der Vorgang `OPTIONS /tasks` tasks liefert eine Antwort auf die Anfrage mit den richtigen CORS-Antwortheadern.

Antwort

Name	Description
`taskIds`	Eine Liste von Aufgaben IDs zum Ausführen von Tests

GET /regions

Description

Mit diesem `GET /regions` Vorgang können Sie die regionalen Ressourceninformationen abrufen, die für die Durchführung eines Tests in dieser Region erforderlich sind.

Antwort

Name	Description
`testId`	Die Regions-ID
`ecsCloudWatchLogGroup`	Der Name der CloudWatch Amazon-Protokollgruppe für die Amazon Fargate-Aufgaben in der Region
`region`	Die Region, in der die Ressourcen in der Tabelle existieren
`subnetA`	Die ID eines der Subnetze in der Region
`subnetB`	Die ID eines der Subnetze in der Region

Name	Description
`taskCluster`	Der Name des AWS Fargate-Clusters in der Region
`taskDefinition`	Der ARN der Aufgabendefinition in der Region
`taskImage`	Der Name des Task-Images in der Region
`taskSecurityGroup`	Die ID der Sicherheitsgruppe in der Region

OPTIONEN /Regionen

Description

Der `OPTIONS /regions` Vorgang liefert eine Antwort auf die Anfrage mit den richtigen CORS-Antwortheadern.

Antwort

Name	Description
`testId`	Die Regions-ID
`ecsCloudWatchLogGroup`	Der Name der CloudWatch Amazon-Protokollgruppe für die Amazon Fargate-Aufgaben in der Region
`region`	Die Region, in der die Ressourcen in der Tabelle existieren
`subnetA`	Die ID eines der Subnetze in der Region
`subnetB`	Die ID eines der Subnetze in der Region
`taskCluster`	Der Name des AWS Fargate-Clusters in der Region
`taskDefinition`	Der ARN der Aufgabendefinition in der Region

Name	Description
`taskImage`	Der Name des Task-Images in der Region
`taskSecurityGroup`	Die ID der Sicherheitsgruppe in der Region

Erhöhen Sie die Container-Ressourcen

Um die Anzahl der gleichzeitigen virtuellen Benutzer (Parallelität) zu erhöhen, die Ihre Auslastungstests simulieren können, müssen Sie die CPU- und Speicherressourcen erhöhen, die jeder Amazon ECS-Aufgabe zugewiesen sind. Dazu müssen Sie eine neue Version der Aufgabendefinition mit höheren Ressourcenlimits erstellen und anschließend die DynamoDB-Konfiguration der Lösung aktualisieren, um die neue Aufgabendefinition für future Testläufe zu verwenden.

Erstellen Sie eine neue Version der Aufgabendefinition

Gehen Sie wie folgt vor, um eine neue Aufgabendefinition mit erhöhten CPU- und Speicherressourcen zu erstellen:

1. Melden Sie sich bei der Amazon Elastic Container Service-Konsole an.
2. Wählen Sie im linken Navigationsmenü Aufgabendefinitionen aus.
3. Aktivieren Sie das Kontrollkästchen neben der Aufgabendefinition, die dieser Lösung entspricht. Zum Beispiel `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*.
4. Wählen Sie Create new revision (Neue Revision erstellen) aus.
5. Führen Sie auf der Seite Neue Revision erstellen die folgenden Aktionen aus:
   a. Ändern Sie unter Task-Größe die Werte Task-Speicher und Task-CPU auf die gewünschten Werte. Höhere Werte ermöglichen mehr gleichzeitige virtuelle Benutzer pro Aufgabe.
   b. Überprüfen Sie unter Containerdefinitionen die Grenzwerte für Hard/Soft Memory. Wenn dieses Limit unter dem von Ihnen gewünschten Speicherplatz liegt, wählen Sie den Container aus.
   c. Gehen Sie im Dialogfeld „Container bearbeiten“ zu Speicherlimits und aktualisieren Sie das feste Limit, sodass es mit Ihrer Aufgabenspeicherzuweisung übereinstimmt oder darunter liegt.
   d. Wählen Sie Aktualisieren aus.
6. Wählen Sie auf der Seite Neue Revision erstellen die Option Erstellen aus.

7. Nachdem die Aufgabendefinition erfolgreich erstellt wurde, notieren Sie den vollständigen ARN der Aufgabendefinition einschließlich der Versionsnummer. Zum Beispiel: `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*: [austauschbar]`<system-generated-versionNumber>`.

Aktualisieren Sie die DynamoDB-Tabelle

Nachdem Sie die neue Version der Aufgabendefinition erstellt haben, müssen Sie die DynamoDB-Tabelle der Lösung aktualisieren, sodass future Testläufe die neue Aufgabendefinition verwenden. Wiederholen Sie diese Schritte für jede AWS-Region, in der Sie die aktualisierte Aufgabendefinition verwenden möchten:

1. Navigieren Sie zur DynamoDB-Konsole.
2. Wählen Sie im linken Navigationsbereich unter Tabellen die Option Elemente durchsuchen aus.
3. Wählen Sie die `scenarios-table` DynamoDB-Tabelle aus, die dieser Lösung zugeordnet ist. Zum Beispiel `[replaceable]` <stackName>`- DLTTest RunnerStorage DLTScenarios Table-. *<system-generated-random-Hash>*
4. Wählen Sie das Element aus, das der Region entspricht, in der Sie die neue Version der Aufgabendefinition erstellt haben. Zum Beispiel `region-[replaceable]` <region-name>`.
5. Suchen Sie im Element-Editor das TaskDefinition-Attribut und aktualisieren Sie seinen Wert mit dem vollständigen Task-Definition-ARN, den Sie im vorherigen Abschnitt aufgezeichnet haben (einschließlich der Versionsnummer).
6. Wählen Sie Änderungen speichern aus.

> **Note**
>
> Die aktualisierte Aufgabendefinition wird nur für neue Testläufe verwendet. Für alle derzeit laufenden oder geplanten Tests wird weiterhin die vorherige Aufgabendefinition verwendet.

Spezifikation der MCP-Tools

Die Distributed Load Testing-Lösung bietet eine Reihe von MCP-Tools, mit denen KI-Agenten mit Testszenarien und Ergebnissen interagieren können. Diese Tools bieten umfassende, abstrakte Funktionen, die auf die Art und Weise abgestimmt sind, wie KI-Agenten Informationen verarbeiten,

sodass sie sich auf Analysen und Erkenntnisse konzentrieren können, anstatt sich auf detaillierte API-Verträge zu konzentrieren.

> **Note**
>
> Alle MCP-Tools bieten nur Lesezugriff auf die Daten der Lösung. Über die MCP-Schnittstelle werden keine Änderungen an Testszenarien oder Konfigurationen unterstützt.

list_scenarios

Description

Das `list_scenarios` Tool ruft eine Liste aller verfügbaren Testszenarien mit grundlegenden Metadaten ab.

Endpoint

`GET /scenarios`

Parameters

Keine

Antwort

Name	Description
`testId`	Eindeutiger Bezeichner für das Testszenario
`testName`	Name des Testszenarios
`status`	Aktueller Status des Testszenarios
`startTime`	Wann der Test erstellt oder zuletzt ausgeführt wurde
`testDescription`	Beschreibung des Testszenarios

get_scenario_details

Description

Das `get_scenario_details` Tool ruft die Testkonfiguration und den letzten Testlauf für ein einzelnes Testszenario ab.

Endpoint

```
GET /scenarios/<test_id>?history=false&results=false
```

Parameter anfordern

`test_id`

- Die eindeutige Kennung für das Testszenario

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

Name	Description
`testTaskConfigs`	Aufgabenkonfiguration für jede Region
`testScenario`	Testdefinition und Parameter
`status`	Aktueller Teststatus
`startTime`	Zeitstempel für den Teststart
`endTime`	Endzeitstempel des Tests (falls abgeschlossen)

list_test_runs

Description

Das `list_test_runs` Tool ruft eine Liste von Testläufen für ein bestimmtes Testszenario ab, sortiert vom neuesten zum ältesten. Gibt maximal 30 Ergebnisse zurück.

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=<limit>
```

oder

```
GET /scenarios/<testid>/testruns/?
limit=30&start_date=<start_date>&end_date=<end_date>
```

Anforderungsparameter

`test_id`

- Der eindeutige Bezeichner für das Testszenario

  Typ: Zeichenfolge

  Erforderlich: Ja

`limit`

- Maximale Anzahl zurückzugebender Testläufe

  Typ: Ganzzahl

  Standard: 20

  Maximum: 30

  Erforderlich: Nein

`start_date`

- ISO 8601-Zeitstempel zum Filtern von Läufen ab einem bestimmten Datum

  Typ: Zeichenfolge (Datums-/Uhrzeitformat)

  Erforderlich: Nein

`end_date`

- Der ISO 8601-Zeitstempel zum Filtern läuft bis zu einem bestimmten Datum

  Typ: Zeichenfolge (Datums-/Uhrzeitformat)

  Erforderlich: Nein

Antwort

Name	Description
`testRuns`	Reihe von Testlaufzusammenfassungen mit Leistungskennzahlen und Perzentilen für jeden Lauf

get_test_run

Description

Das `get_test_run` Tool ruft detaillierte Ergebnisse für einen einzelnen Testlauf mit regionalen und Endpunktaufschlüsselungen ab.

Endpoint

`GET /scenarios/<testid>/testruns/<testrunid>`

Anforderungsparameter

`test_id`

- Die eindeutige Kennung für das Testszenario

  Typ: Zeichenfolge

  Erforderlich: Ja

`test_run_id`

- Die eindeutige Kennung für den spezifischen Testlauf

  Typ: Zeichenfolge

Erforderlich: Ja

Antwort

Name	Description
`results`	Vollständige Testlaufdaten, einschließlich Aufschlüsselung der regionalen Ergebnisse, endpunktspezifischer Metriken, Leistungsperzentile (p50, p90, p95, p99), Erfolgs- und Fehlschlagzahlen, Reaktionszeiten und Latenz sowie der für den Testlauf verwendeten Testkonfiguration

get_latest_test_run

Description

Das `get_latest_test_run` Tool ruft den letzten Testlauf für ein bestimmtes Testszenario ab.

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=1
```

> **Note**
>
> Die Ergebnisse werden anhand eines Global Secondary Index (GSI) nach Zeit sortiert, sodass sichergestellt wird, dass der letzte Testlauf zurückgegeben wird.

Anforderungsparameter

`test_id`

- Die eindeutige Kennung für das Testszenario

  Typ: Zeichenfolge

Erforderlich: Ja

Antwort

Name	Description
`results`	Aktuelle Testlaufdaten mit demselben Format wie `get_test_run`

get_baseline_test_run

Description

Das `get_baseline_test_run` Tool ruft den Basistestlauf für ein bestimmtes Testszenario ab. Die Baseline wird zu Leistungsvergleichszwecken verwendet.

Endpoint

```
GET /scenarios/<test_id>/baseline
```

Anforderungsparameter

`test_id`

- Die eindeutige Kennung für das Testszenario

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

Name	Description
`baselineData`	Basisdaten des Testlaufs zu Vergleich szwecken, einschließlich aller Metriken und

Name	Description
	Konfigurationen aus dem angegebenen Baselinelauf

get_test_run_artifacts

Description

Das `get_test_run_artifacts` Tool ruft Amazon S3 S3-Bucket-Informationen für den Zugriff auf Testartefakte wie Protokolle, Fehlerdateien und Ergebnisse ab.

Endpoint

```
GET /scenarios/<testid>/testruns/<testrunid>
```

Anforderungsparameter

`test_id`

- Die eindeutige Kennung für das Testszenario

  Typ: Zeichenfolge

  Erforderlich: Ja

`test_run_id`

- Die eindeutige Kennung für den spezifischen Testlauf

  Typ: Zeichenfolge

  Erforderlich: Ja

Antwort

Name	Description
`bucketName`	Name des S3-Buckets, in dem Artefakte gespeichert werden

Name	Description
`testRunPath`	Pfadpräfix für den aktuellen Artefaktspeicher (Version 4.0+)
`testScenarioPath`	Pfadpräfix für älteren Artefaktspeicher (vor Version 4.0)

> **Note**
>
> Alle MCP-Tools nutzen bestehende API-Endpunkte. Zur Unterstützung der MCP-Funktionalität APIs sind keine Änderungen am Basisprodukt erforderlich.

Referenz

Dieser Abschnitt enthält Informationen zur Datenerfassung, Hinweise auf verwandte Ressourcen und eine Liste der Entwickler, die zu dieser Lösung beigetragen haben.

Datenerfassung

Diese Lösung sendet Betriebsmetriken (die „Daten“) über die Verwendung dieser Lösung an AWS. Wir verwenden diese Daten, um besser zu verstehen, wie Kunden diese Lösung und die damit verbundenen Dienstleistungen und Produkte nutzen. Die Erfassung dieser Daten durch AWS unterliegt der AWS-Datenschutzerklärung.

Mitwirkende

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dmitri López
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhil Reddy
- Simon Kroll
- Ahern Knox
- Ian Downard
- Owen Brady
- Jim Thario
- Thyag Ramachandran
- Yang Qin
- James Wang

Glossar

In diesem Glossar werden die im Implementierungsleitfaden für Distributed Load Testing on AWS verwendeten Akronyme und Abkürzungen definiert.

Technische Protokolle und Formate

AGPL

Allgemeine öffentliche Lizenz von Affero. Eine von K6 verwendete Open-Source-Softwarelizenz.

API

Schnittstelle zur Anwendungsprogrammierung. Eine Reihe von Protokollen und Tools zum Erstellen von Softwareanwendungen und zum Ermöglichen der Kommunikation zwischen verschiedenen Systemen.

CLI

Befehlszeilenschnittstelle. Eine textbasierte Oberfläche für die Interaktion mit Software und Betriebssystemen.

KERNE

Quellenübergreifende gemeinsame Nutzung von Ressourcen. Eine Sicherheitsfunktion, die Webanwendungen, die an einem Ursprung ausgeführt werden, den Zugriff auf Ressourcen von einem anderen Ursprung ermöglicht oder einschränkt.

CSV

Durch Kommas getrennte Werte. Ein Dateiformat, das zum Speichern von Tabellendaten im Klartext verwendet wird und häufig für den Datenexport verwendet wird.

gRPC

gRPC Remote Procedure Call. Ein leistungsstarkes Open-Source-Framework für Remote-Prozeduraufrufe.

HTTP

Hypertext-Übertragungsprotokoll. Das Basisprotokoll, das für die Übertragung von Daten im World Wide Web verwendet wird.

HTTPS

HTTP Secure. Eine Erweiterung von HTTP, die Verschlüsselung für die sichere Kommunikation über ein Netzwerk verwendet.

JSON

JavaScript Objektnotation. Ein leichtes Datenaustauschformat, das für Menschen leicht zu lesen und zu schreiben und für Maschinen einfach zu analysieren und zu generieren ist.

JWT

JSON-Webtoken. Ein kompaktes, URL-sicheres Mittel zur Darstellung von Ansprüchen, die zur Authentifizierung und Autorisierung zwischen zwei Parteien übertragen werden sollen.

OAuth

Öffnen Sie die Autorisierung. Ein offener Standard für die Zugriffsdelegierung, der häufig für die tokenbasierte Authentifizierung und Autorisierung verwendet wird.

REST

Repräsentativer Staatstransfer. Ein Architekturstil für den Entwurf von Netzwerkanwendungen mit zustandsloser Kommunikation und Standard-HTTP-Methoden.

SSE

Vom Server gesendete Ereignisse. Eine Server-Push-Technologie, die es einem Client ermöglicht, automatische Updates von einem Server über eine HTTP-Verbindung zu empfangen.

Benutzeroberfläche

Benutzerschnittstelle. Die visuellen Elemente und Steuerelemente, über die Benutzer mit Softwareanwendungen interagieren.

URL

Einheitlicher Resource Locator. Die Adresse, die für den Zugriff auf Ressourcen im Internet verwendet wird.

XML

Erweiterbare Auszeichnungssprache. Eine Auszeichnungssprache, die Regeln für die Kodierung von Dokumenten in einem Format definiert, das sowohl für Menschen als auch für Maschinen lesbar ist.

Begriffe zum Testen und zur Datenbank

FTP

Dateiübertragungsprotokoll. Ein Standard-Netzwerkprotokoll, das für die Übertragung von Dateien zwischen einem Client und einem Server verwendet wird.

GSI

Globaler sekundärer Index. Eine DynamoDB-Funktion, mit der Sie Daten mit einem alternativen Schlüssel abfragen können.

JDBC

Java-Datenbankkonnektivität. Eine Java-API zum Verbinden und Ausführen von Abfragen mit Datenbanken.

JMS

Java-Nachrichtendienst. Eine Java-API zum Senden von Nachrichten zwischen zwei oder mehr Clients.

TPS

Transaktionen pro Sekunde. Ein Maß für die Anzahl der Transaktionen, die ein System in einer Sekunde verarbeiten kann.

AWS- und Systembedingungen

ARN

Amazon-Ressourcenname. Eine eindeutige Kennung für AWS-Ressourcen, die zur Spezifizierung von Ressourcen für alle AWS-Services verwendet wird.

ISO

Internationale Organisation für Standardisierung. Eine unabhängige Nichtregierungsorganisation, die internationale Standards entwickelt. In diesem Handbuch wird auf das ISO 8601-Zeitstempelformat verwiesen.

SLA

Vereinbarung zum Servicelevel. Eine Verpflichtung zwischen einem Dienstleister und einem Kunden, die das erwartete Serviceniveau definiert.

UUID

Universell eindeutiger Bezeichner. Eine 128-Bit-Nummer, die zur eindeutigen Identifizierung von Informationen in Computersystemen verwendet wird.

vCPU

Virtuelle Zentraleinheit. Ein virtueller Prozessor, der einer virtuellen Maschine oder einem Container zugewiesen ist und einen Teil der Rechenleistung der physischen CPU darstellt.

Bedingungen für den Belastungstest

concurrency

Die Anzahl gleichzeitiger virtueller Benutzer pro Aufgabe. Dieser Parameter steuert, wie viele simulierte Benutzer jede Fargate-Aufgabe während eines Auslastungstests generiert.

regionaler Stapel

Ein CloudFormation Stack, der in einer AWS-Region bereitgestellt wird, um eine Testinfrastruktur für Lasttests in mehreren Regionen bereitzustellen.

Anzahl der Aufgaben

Die Anzahl der Fargate-Container (Aufgaben), die zur Ausführung eines Testszenarios gestartet wurden. Die generierte Gesamtlast entspricht der Anzahl der Aufgaben multipliziert mit der Parallelität.

Testszenario

Ein konfigurierter Auslastungstest, der Testtyp, Zielendpunkte, Anzahl der Aufgaben, Parallelität, Dauer und andere Parameter umfasst.

Überarbeitungen

Besuchen Sie CHANGELOG.md in unserem GitHub Repository, um versionsspezifische Verbesserungen und Korrekturen nachzuverfolgen.

Hinweise

Kunden sind dafür verantwortlich, Ihre eigene unabhängige Bewertung der Informationen in diesem Dokument vorzunehmen. Dieses Dokument: (a) dient nur zu Informationszwecken, (b) stellt aktuelle Produktangebote und Praktiken von AWS dar, die ohne vorherige Ankündigung geändert werden können, und (c) stellt keine Verpflichtungen oder Zusicherungen von AWS und seinen verbundenen Unternehmen, Lieferanten oder Lizenzgebern dar. AWS-Produkte oder -Services werden „wie sie sind“ ohne ausdrückliche oder stillschweigende Garantien, Zusicherungen oder Bedingungen jeglicher Art bereitgestellt. Die Verantwortlichkeiten und Verbindlichkeiten von AWS gegenüber seinen Kunden werden durch AWS-Verträge geregelt. Dieses Dokument ist weder Teil einer Vereinbarung zwischen AWS und seinen Kunden noch ändert es diese.

Distributed Load Testing auf AWS ist unter den Bedingungen der Apache License Version 2.0 lizenziert, die bei The Apache Software Foundation erhältlich ist.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.