aws

Entwicklerhandbuch

Amazon API Gateway

Amazon API Gateway: Entwicklerhandbuch

Table of Contents

Was ist Amazon API Gateway?

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

API Gateway erstellt RESTful-APIs, die:

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

Weitere Informationen über API Gateway-REST-APIs und -HTTP-APIs finden Sie unter the section called “Auswahl zwischen REST-APIs und HTTP-APIs ”, Arbeiten mit HTTP-APIs, the section called “API Gateway zur Erstellung von REST-APIs verwenden” und the section called “Entwickeln”.

API Gateway erstellt WebSocket APIs, die:

- Halten Sie sich an das WebSocketProtokoll, das eine statusbehaftete Vollduplex-Kommunikation zwischen Client und Server ermöglicht.
- eingehende Nachrichten basierend auf dem Inhalt der Nachricht weiterleiten.

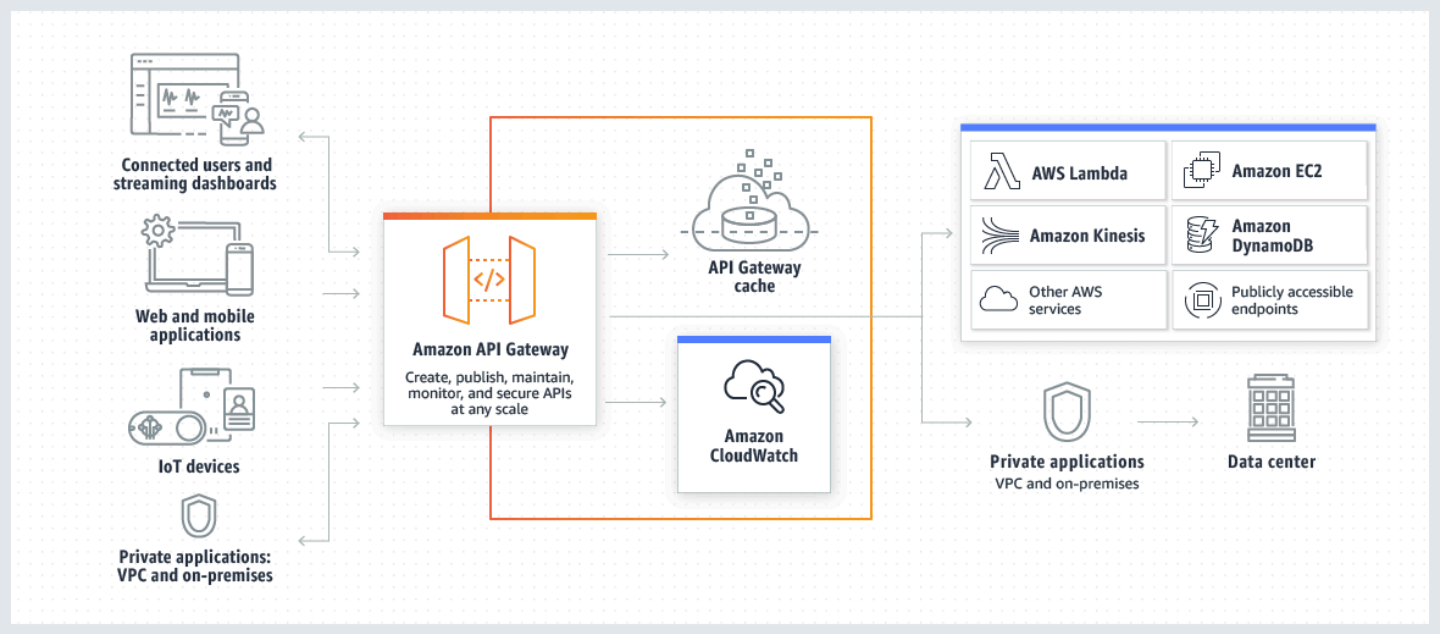
Weitere Informationen zu API Gateway WebSocket Gateway-APIs finden Sie unter the section called “Verwenden Sie API Gateway, um WebSocket APIs zu erstellen” undthe section called “Über WebSocket APIs”.

Themen

- Teil der AWS serverlosen Infrastruktur
- So steigen Sie in Amazon API Gateway ein:
- Amazon API Gateway-Konzepte
- Auswahl zwischen REST-APIs und HTTP-APIs
- Erste Schritte mit der REST-API-Konsole

Architektur von API Gateway

Das folgende Diagramm zeigt die API Gateway-Architektur.



Dieses Diagramm veranschaulicht, wie die APIs, die Sie in Amazon API Gateway erstellen, Ihnen oder Ihren Entwicklerkunden eine integrierte und konsistente Entwicklererfahrung für die Erstellung von Serverless AWS -Anwendungen bieten. API Gateway handhabt sämtliche Aufgaben im Zusammenhang mit der Annahme und Verarbeitung von Hunderttausenden gleichzeitiger API-Aufrufe. Zu diesen Aufgaben gehören die Verwaltung des Datenverkehrs, Autorisierung und Zugriffskontrolle, Überwachung und Verwaltung der API-Version.

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

Funktionen von API Gateway

Amazon API Gateway bietet zum Beispiel die folgenden Funktionen:

- Support für Stateful (WebSocket) und Stateless (HTTP und REST) APIs.
- Leistungsstarke, flexible Authentifizierungsmechanismen wie AWS Identity and Access Management Richtlinien, Lambda-Autorisierungsfunktionen und Amazon Cognito Cognito-Benutzerpools.
- Canary-Release-Bereitstellungen für sichere fortlaufende Änderungen
- CloudTrailProtokollierung und Überwachung der API-Nutzung und API-Änderungen.
- CloudWatch Zugriffs- und Ausführungsprotokollierung, einschließlich der Möglichkeit, Alarme einzustellen. Weitere Informationen finden Sie unter the section called “CloudWatch Metriken” und the section called “Metriken”.
- Möglichkeit, AWS CloudFormation Vorlagen zu verwenden, um die API-Erstellung zu ermöglichen. Weitere Informationen finden Sie unter Referenz der Amazon API Gateway-Ressourcentypen und Referenz der Amazon API Gateway V2-Ressourcentypen.
- Unterstützung benutzerdefinierter Domänennamen
- AWS WAF-Integration zum Schutz Ihrer APIs vor vom Web ausgehenden Übergriffen
- AWS X-Ray-Integration für ein besseres Verständnis und die Analyse leistungsbezogener Latenzen

Eine vollständige Liste der API Gateway-Funktionsversionen finden Sie unter Dokumentverlauf.

Anwendungsfälle für API Gateway

Themen

- API Gateway zur Erstellung von REST-APIs verwenden
- API Gateway zur Erstellung von HTTP-APIs verwenden
- Verwenden Sie API Gateway, um WebSocket APIs zu erstellen
- Von wem wird API Gateway verwendet?

API Gateway zur Erstellung von REST-APIs verwenden

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

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

Die App benötigt keine Informationen dazu, wo auf dem Backend die angeforderten Daten gespeichert und abgerufen werden. Bei API Gateway-REST-APIs wird das Frontend durch Methoden-Anfragen und Methoden-Antworten gekapselt. Die API arbeitet über Integrationsanforderungen und Integrationsantworten mit dem Backend zusammen.

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

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

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

- Unterstützung für die Generierung von SDKs und die Erstellung einer API-Dokumentation mit API Gateway-Erweiterungen für OpenAPI
- Drosselung von HTTP-Anforderungen

API Gateway zur Erstellung von HTTP-APIs verwenden

Mit HTTP-APIs können Sie REST-fähige APIs mit geringerer Latenzzeit und zu geringeren Kosten als REST-APIs erstellen.

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

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

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

Weitere Informationen hierzu finden Sie unter the section called “Auswahl zwischen REST-APIs und HTTP-APIs ”.

Verwenden Sie API Gateway, um WebSocket APIs zu erstellen

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

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

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

- Chat-Anwendungen
- Echtzeit-Dashboards, wie z. B. Aktienticker
- Echtzeit-Warnungen und -Benachrichtigungen

API Gateway bietet WebSocket API-Verwaltungsfunktionen wie die folgenden:

- Überwachung und Drosselung von Verbindungen und Nachrichten
- Wird verwendet AWS X-Ray , um Nachrichten auf ihrem Weg durch die APIs zu Back-End-Diensten zu verfolgen
- Einfache Integration in HTTP/HTTPS-Endpunkte

Von wem wird API Gateway verwendet?

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

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

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

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

API Gateway-API erstellen und verwalten

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

Als API-Entwickler können Sie eine API erstellen und verwalten, indem Sie die unter Erste Schritte mit API Gateway beschriebene API Gateway-Konsole verwenden oder die API-Referenzen aufrufen. Es gibt mehrere Möglichkeiten, diese API aufzurufen. Dazu gehören die Verwendung von AWS Command Line Interface (AWS CLI) oder die Verwendung eines SDK. AWS Außerdem ist die API-Erstellung auch mittels AWS CloudFormation -Vorlagen oder (im Falle von REST- und HTTP-APIs) API Gateway-Erweiterungen für OpenAPI arbeiten möglich.

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

API Gateway-API aufrufen

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

Auf API Gateway zugreifen

Sie können wie folgt auf Amazon API Gateway zugreifen:

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

Teil der AWS serverlosen Infrastruktur

Zusammen mit AWS LambdaAPI Gateway bildet das API Gateway den anwendungsorientierten Teil der AWS serverlosen Infrastruktur. Weitere Informationen zu den ersten Schritten mit Serverless-Technologie finden Sie im Serverless-Entwicklerhandbuch.

Damit eine App öffentlich verfügbare AWS Dienste aufruft, können Sie Lambda verwenden, um mit den erforderlichen Diensten zu interagieren und Lambda-Funktionen über API-Methoden in

API Gateway verfügbar zu machen. AWS Lambda führt Ihren Code auf einer hochverfügbaren Computerinfrastruktur aus. Es übernimmt die erforderliche Ausführung und Verwaltung der Datenverarbeitungsressourcen. Um serverlose Anwendungen zu ermöglichen, unterstützt API Gateway optimierte Proxyintegrationen mit AWS Lambda und HTTP-Endpunkten.

So steigen Sie in Amazon API Gateway ein:

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

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

Amazon API Gateway-Konzepte

API Gateway

API Gateway ist ein AWS Dienst, der Folgendes unterstützt:

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

API Gateway-REST-API

Eine Sammlung von HTTP-Ressourcen und -Methoden, die in Backend-HTTP-Endpunkte, Lambda-Funktionen oder andere Dienste integriert sind. AWS Sie können diese Sammlung in einer oder mehreren Stufen bereitstellen. In der Regel werden API-Ressourcen in einer Ressourcenstruktur organisiert, die der Anwendungslogik entspricht. Jede API-Ressource kann eine oder mehrere API-Methoden offenlegen, die eindeutige HTTP-Verben haben, die von API Gateway unterstützt werden. Weitere Informationen finden Sie unter the section called “Auswahl zwischen REST-APIs und HTTP-APIs ”.

API Gateway-HTTP-API

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

WebSocket API-Gateway-API

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

API-Bereitstellung

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

API-Developer

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

API-Endpunkt

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

- Edge-optimierter API-Endpunkt
- Privater API-Endpunkt
- Regionaler API-Endpunkt

API-Schlüssel

Eine alphanumerische Zeichenfolge, die API Gateway verwendet, um einen App-Entwickler zu identifizieren, der Ihre REST oder WebSocket API verwendet. API Gateway kann API-Schlüssel für Sie erstellen. Alternativ können Sie Schlüssel aus einer CSV-Datei importieren. Sie können API-Schlüssel zusammen mit Lambda-Genehmigern oder Nutzungsplänen verwenden, um den Zugriff auf Ihre APIs zu steuern.

Siehe API-Endpunkte.

API-Eigentümer

Siehe API-Developer.

API-Stufe

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

App-Developer

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

Rückruf-URL

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

Entwicklerportal

Eine Anwendung, mit der Ihre Kunden Ihre API-Produkte (API Gateway-Nutzungspläne) registrieren, erkennen und abonnieren, ihre API-Schlüssel verwalten und ihre Nutzungsmetriken für Ihre APIs anzeigen können.

Edge-optimierter API-Endpunkt

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

Siehe API-Endpunkte.

Integrationsanforderung

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

Integrationsantwort

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

Zuweisungsvorlage

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

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

Methodenanforderung

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

Methodenantwort

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

Pseudointegration

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

Model

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

kann. Für die Erstellung einer Zuweisungsvorlage ist ein Modell zwar hilfreich, aber nicht erforderlich.

Privates API

Siehe Privater API-Endpunkt.

Privater API-Endpunkt

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

Private Integration

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

Proxy-Integration

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

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

In REST-APIs wird die Proxy-Integration meist mit einer Proxy-Ressource verwendet, die durch eine gierige Pfadvariable (z. B. `{proxy+}`) und die Catch-all-Methode `ANY` dargestellt wird.

Quick Create

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

Regionaler API-Endpunkt

Der Hostname einer API, die in der angegebenen Region bereitgestellt wird und Clients wie EC2-Instances in derselben Region bedienen soll. AWS API-Anfragen werden direkt an die

regionsspezifische API Gateway gerichtet, ohne dass eine Verteilung erforderlich ist. CloudFront Bei Anfragen innerhalb einer Region umgeht ein regionaler Endpunkt den unnötigen Roundtrip zu einer Distribution. CloudFront

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

Siehe API-Endpunkte.

Route

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

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

Routenanforderung

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

Routenantwort

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

Nutzungsplan

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

WebSocket Verbindung

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

Auswahl zwischen REST-APIs und HTTP-APIs

REST-APIs und HTTP-APIs sind beide RESTful-API-Produkte. REST-APIs unterstützen mehr Funktionen als HTTP-APIs. HTTP-APIs sind mit minimalen Funktionen ausgestattet, sodass sie zu einem niedrigeren Preis angeboten werden können. Wählen Sie REST-APIs, wenn Sie Funktionen wie beispielsweise API-Schlüssel, clientbasierte Drosselung, Anforderungsvalidierung, AWS WAF -Integration oder private API-Endpunkte benötigen. Wählen Sie HTTP-APIs aus, wenn Sie die in REST-APIs enthaltenen Funktionen nicht benötigen.

In den folgenden Abschnitten werden Kernfunktionen zusammengefasst, die in REST-APIs und HTTP-APIs verfügbar sind.

Endpunkttyp

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

Endpunkttypen	REST-API	HTTP-API
Edge-optimiert	✓	
Regional	✓	✓
Privat	✓	

Sicherheit

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

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

Autorisierung

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre API. Weitere Informationen finden Sie unter the section called “Zugriffskontrolle” und the section called “Zugriffskontrolle”.

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

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

[2] Sie können einen Lambda-Genehmiger verwenden, um JWTs für REST-APIs zu validieren.

API-Management

Wählen Sie REST-APIs, wenn Sie API-Verwaltungsfunktionen wie beispielsweise API-Schlüssel und clientbasierte Begrenzungen benötigen. Weitere Informationen finden Sie unter the section

called “Verteilen”, the section called “Benutzerdefinierte Domänennamen” und the section called “Benutzerdefinierte Domänennamen”.

Features	REST-API	HTTP-API
Benutzerdefinierte Domänen	✓	✓
API-Schlüssel	✓	
Clientbasierte Ratenbegrenzung	✓	
Clientbasierte Nutzungsdrosselung	✓	

Entwicklung

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

Features	REST-API	HTTP-API
CORS-Konfiguration	✓	✓
Testaufrufe	✓	
Caching	✓	
Benutzergesteuerte Bereitstellungen	✓	✓
Automatische Bereitstellungen		✓
Benutzerdefinierte Gateway-Antworten	✓	
Canary-Release-Bereitstellungen	✓	

Features	REST-API	HTTP-API
Anforderungsvalidierung	✓	
Transformation von Anfragepa rametern	✓	✓
Transformation von Anforderu ngstext	✓	

Überwachen

API Gateway unterstützt verschiedene Optionen zum Protokollieren von API-Anforderungen und zur Überwachung Ihrer APIs. Weitere Informationen finden Sie unter the section called “Monitor” und the section called “Monitor”.

Funktion	REST-API	HTTP-API
CloudWatch Amazon-Me triken	✓	✓
Zugriff auf Logs auf CloudWatch Logs	✓	✓
Zugriffsprotokolle auf Amazon Data Firehose	✓	
Ausführungsprotokolle	✓	
AWS X-Ray Rückverfolgung	✓	

Integrationen

Integrationen verbinden Ihre API-Gateway-API mit Backend-Ressourcen. Weitere Informationen finden Sie unter the section called “Integrationen” und the section called “Integrationen”.

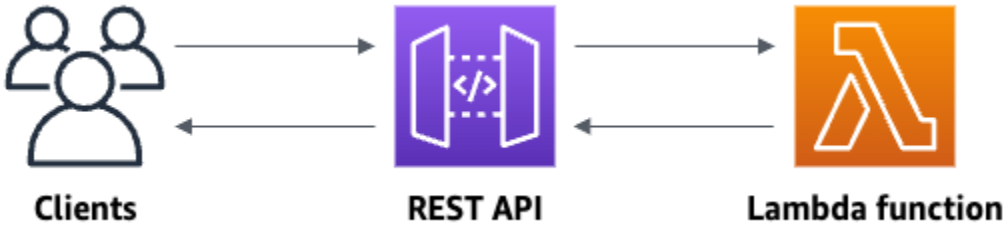
Funktion	REST-API	HTTP-API
Öffentliche HTTP-Endpunkte	✓	✓
AWS Dienstleistungen	✓	✓
AWS Lambda Funktionen	✓	✓
Private Integrationen mit Network Load Balancers	✓	✓
Private Integrationen mit Application Load Balancers		✓
Private Integrationen mit AWS Cloud Map		✓
Pseudointegrationen	✓	

Erste Schritte mit der REST-API-Konsole

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

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

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



Um diese Übung abzuschließen, benötigen Sie einen AWS-Konto und einen AWS Identity and Access Management (IAM-) Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter Voraussetzungen für die ersten Schritte mit API Gateway.

Themen

- Schritt 1: Erstellen einer Lambda-Funktion
- Schritt 2: Erstellen einer REST-API
- Schritt 3: Erstellen einer Lambda-Proxy-Integration
- Schritt 4: Bereitstellen Ihrer API
- Schritt 5: Aufrufen Ihrer API
- (Optional) Schritt 6: Bereinigen

Schritt 1: Erstellen einer Lambda-Funktion

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

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

Eine Lambda-Funktion erstellen

1. Melden Sie sich bei der Lambda-Konsole unter https://console.aws.amazon.com/lambda an.
2. Wählen Sie Create function (Funktion erstellen).
3. Geben Sie unter Basic Information (Grundlegende Informationen) für Function name (Funktionsname) **`my-function`** ein.
4. Wählen Sie Funktion erstellen.

Der standardmäßige Lambda-Funktionscode sollte wie folgt aussehen:

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

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

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

Schritt 2: Erstellen einer REST-API

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

So erstellen Sie eine REST-API

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

Schritt 3: Erstellen einer Lambda-Proxy-Integration

Als Nächstes erstellen Sie eine API-Methode für Ihre REST-API in der Root-Ressource (`/`) und integrieren die Methode mithilfe einer Proxy-Integration in Ihre Lambda-Funktion. Bei einer Lambda-

Proxy-Integration leitet API Gateway die eingehende Anfrage vom Client direkt an die Lambda-Funktion weiter.

So erstellen Sie eine Lambda-Proxy-Integration

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

Schritt 4: Bereitstellen Ihrer API

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

Stellen Sie Ihre API bereit

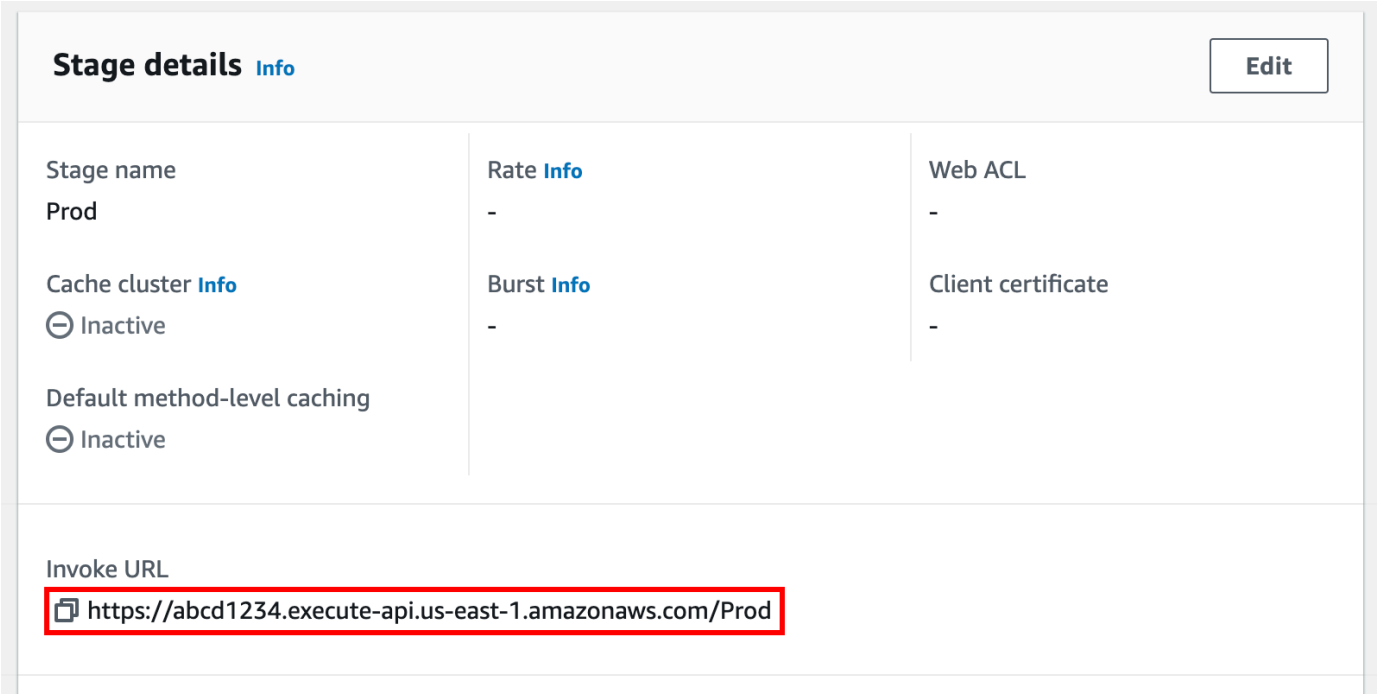
1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **`Prod`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

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

Schritt 5: Aufrufen Ihrer API

So rufen Sie Ihre API auf

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



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

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

   Ihr Browser sendet eine `GET`-Anforderung an die API.

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

(Optional) Schritt 6: Bereinigen

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

So löschen Sie Ihre REST-API

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

So löschen Sie Ihre Lambda-Funktion

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

So löschen Sie die Protokollgruppe Ihrer Lambda-Funktion

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

So löschen Sie die Ausführungsrolle Ihrer Lambda-Funktion:

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

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

Voraussetzungen für die ersten Schritte mit API Gateway

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

Melde dich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

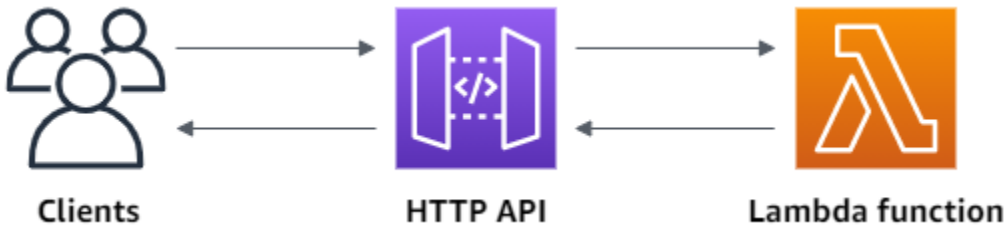
Erste Schritte mit API Gateway

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

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

> **Note**
>
> In dieser Übung wird eine HTTP-API verwendet. API-Gateway unterstützt auch REST-APIs, die mehr Funktionen enthalten. Eine Anleitung zur Verwendung einer REST-API finden Sie unterthe section called “Erste Schritte mit der REST-API-Konsole”.
> Weitere Informationen zum Unterschied zwischen HTTP-APIs und REST-APIs finden Sie unterthe section called “Auswahl zwischen REST-APIs und HTTP-APIs ”.

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



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

Themen

- Schritt 1: Erstellen einer Lambda-Funktion
- Schritt 2: HTTP-API erstellen
- Schritt 3: Ihre API testen

Schritt 1: Erstellen einer Lambda-Funktion

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

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

So erstellen Sie eine Lambda-Funktion:

1. Melden Sie sich bei der Lambda-Konsole unter https://console.aws.amazon.com/lambda an.
2. Wählen Sie Create function (Funktion erstellen).
3. Geben Sie für Function name (Funktionsname) **`my-function`** ein.
4. Wählen Sie Create function (Funktion erstellen).

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

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

Der standardmäßige Lambda-Funktionscode sollte wie folgt aussehen:

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

Schritt 2: HTTP-API erstellen

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

mehr Funktionen als HTTP-APIs, aber wir benötigen diese Funktionen für diese Übung nicht. HTTP-APIs sind mit minimalen Funktionen konzipiert, sodass sie zu einem niedrigeren Preis angeboten werden können. WebSocket APIs unterhalten persistente Verbindungen mit Clients für die Vollduplex-Kommunikation, was in diesem Beispiel nicht erforderlich ist.

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

So erstellen Sie eine HTTP-API

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

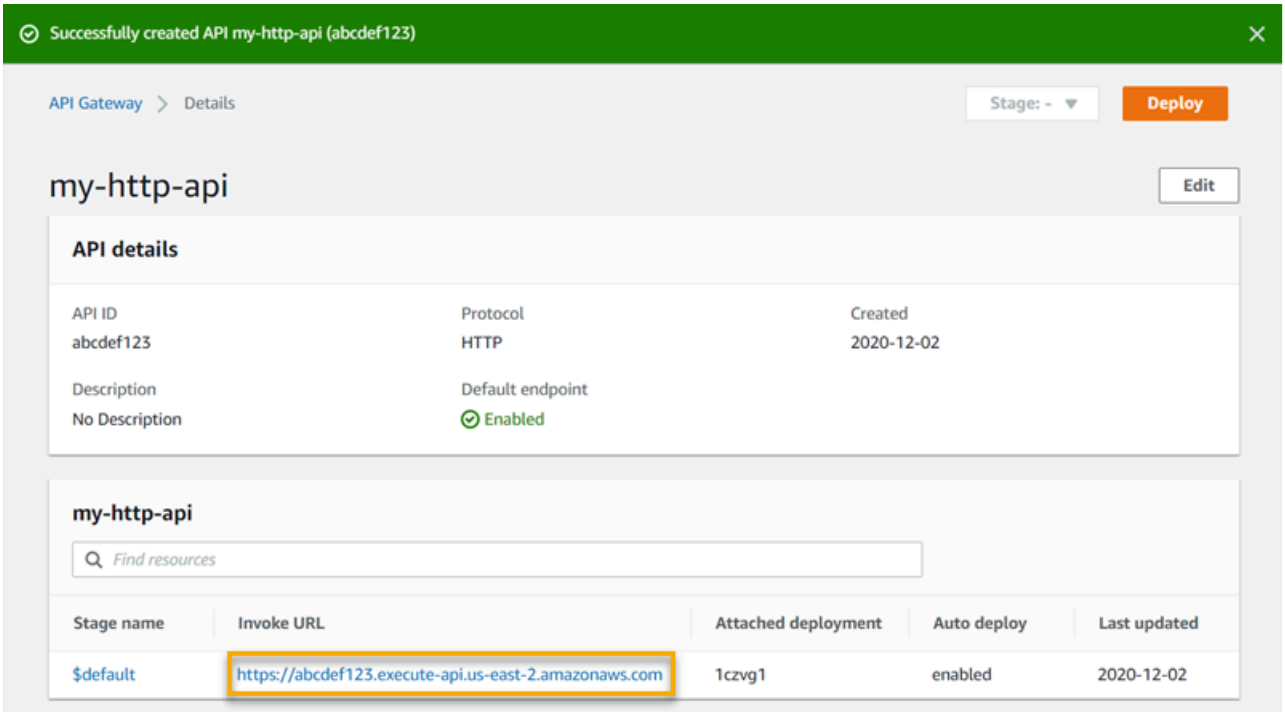
Jetzt haben Sie eine HTTP-API mit einer Lambda-Integration erstellt, die bereit ist, Anfragen von Clients zu erhalten.

Schritt 3: Ihre API testen

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

So testen Sie Ihre API

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



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

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

   Ihr Browser sendet eine `GET`-Anforderung an die API.
5. Verifizieren Sie die Antwort Ihrer API. Sie sollten den Text `"Hello from Lambda!"` in Ihrem Browser sehen.

(Optional) Schritt 4: Bereinigen

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

So löschen Sie eine HTTP-API

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

So löschen Sie eine Lambda-Funktion

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

So löschen Sie die Protokollgruppe einer Lambda-Funktion

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

So löschen Sie die Ausführungsrolle einer Lambda-Funktion

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

Sie können die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie AWS CloudFormation oder AWS SAM verwenden. Unter AWS CloudFormation -Beispielvorlagen finden Sie Beispiele für AWS CloudFormation -Vorlagen.

Nächste Schritte

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

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

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

Hilfe zu Amazon API Gateway aus der Community erhalten Sie im API Gateway-Diskussionsforum. Wenn Sie dieses Forum betreten, müssen Sie sich AWS möglicherweise anmelden.

Wenn Sie direkt Hilfe zu API Gateway erhalten möchten AWS, sehen Sie sich die Support-Optionen auf der AWS Support-Seite an.

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

Amazon API Gateway-Tutorials und -Workshops

Die folgenden Tutorials und Workshops bieten praktische Übungen, die Ihnen beim Kennenlernen von API Gateway helfen.

REST API-Tutorials

- Wählen Sie ein AWS Lambda Integrations-Tutorial
- Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels
- Wählen Sie ein Tutorial zur HTTP-Integration
- Tutorial: REST-API mit privater API Gateway-Integration erstellen
- Tutorial: Erstellen Sie eine API-Gateway-REST-API mit AWS Integration
- Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration
- Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API Gateway
- Tutorial: REST-API als Amazon Kinesis-Proxy in API Gateway erstellen
- Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI
- Tutorial: Erstellen Sie eine private REST-API

HTTP API-Tutorials

- Tutorial: Erstellen einer CRUD-API mit Lambda und DynamoDB
- Tutorial: Aufbau einer HTTP-API mit privater Integration in einen Amazon-ECS-Service

WebSocket API-Tutorials

- Tutorial: Eine serverlose Chat-App mit einer WebSocket API, Lambda und DynamoDB erstellen

Workshops

- Erstellen einer Serverless-Webanwendung
- CI/CD für Serverless-Anwendungen
- Workshop für Serverless-Sicherheit
- Serverless-Identitätsverwaltung, -Authentifizierung und -Autorisierung

- Der Amazon-API-Gateway-Workshop

Amazon API Gateway-REST-API-Tutorials

Die folgenden Tutorials bieten praktische Übungen, die Ihnen beim Kennenlernen von API Gateway REST APIs helfen.

Themen

- Wählen Sie ein AWS Lambda Integrations-Tutorial
- Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels
- Wählen Sie ein Tutorial zur HTTP-Integration
- Tutorial: REST-API mit privater API Gateway-Integration erstellen
- Tutorial: Erstellen Sie eine API-Gateway-REST-API mit AWS Integration
- Tutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration
- Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API Gateway
- Tutorial: REST-API als Amazon Kinesis-Proxy in API Gateway erstellen
- Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI
- Tutorial: Erstellen Sie eine private REST-API

Wählen Sie ein AWS Lambda Integrations-Tutorial

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

Bei der Lambda-Proxyintegration kann die Eingabe für die Lambda-Funktion als eine beliebige Kombination von Anforderungsheadern, Pfadvariablen, Abfragezeichenfolgenparametern, Hauptteil- und API-Konfigurationsdaten ausgedrückt werden. Sie müssen nur eine Lambda-Funktion auswählen. API Gateway konfiguriert die Integrationsanforderung und Integrationsantwort für Sie. Nach der Einrichtung kann sich Ihre API-Methode weiterentwickeln, ohne die vorhandenen Einstellungen zu ändern. Dies ist möglich, weil die Back-End-Lambda-Funktion die eingehenden Anforderungsdaten analysiert und mit dem Client antwortet.

Bei der Lambda-Nicht-Proxy-Integration müssen Sie sicherstellen, dass die Eingabe an die Lambda-Funktion als Payload der Integrationsanforderung geliefert wird. Sie müssen alle vom Client als

Anforderungsparameter bereitgestellten Eingabedaten dem richtigen Text der Integrationsanfrage zuordnen. Möglicherweise müssen Sie auch den vom Client bereitgestellten Anfragetextkörper in ein Format umwandeln, das von der Lambda-Funktion erkannt wird.

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

Themen

- Tutorial: Hello World REST-API mit Lambda-Proxy-Integration erstellen
- Tutorial: API Gateway REST-API mit Lambda Nicht-Proxy-Integration erstellen
- Tutorial: API Gateway REST-API mit kontenübergreifender Lambda-Proxy-Integration erstellen

Tutorial: Hello World REST-API mit Lambda-Proxy-Integration erstellen

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

In diesem Tutorial führen Sie folgende Aufgaben aus:

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

Themen

- Erstellen Sie eine "Hello, World!"- Lambda-Funktion
- Erstellen Sie eine "Hello, World!"- API
- Bereitstellen und Testen der API

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

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

1. Melden Sie sich bei der Lambda-Konsole unter https://console.aws.amazon.com/lambda an.
2. Wählen Sie in der AWS Navigationsleiste eine aus AWS-Region.

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

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

   Node.js

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

```
        if (body.greeter && body.greeter !== "") {
            greeter = body.greeter;
        }
    } else if (event.queryStringParameters &&
 event.queryStringParameters.greeter && event.queryStringParameters.greeter !==
 "") {
        greeter = event.queryStringParameters.greeter;
    } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
 event.multiValueHeaders.greeter != "") {
        greeter = event.multiValueHeaders.greeter.join(" and ");
    } else if (event.headers && event.headers.greeter && event.headers.greeter !
= "") {
        greeter = event.headers.greeter;
    }

    res.body = "Hello, " + greeter + "!";
    callback(null, res);
};
```

Python

```
import json


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

    greeter = 'World'

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

    try:
        if (event['multiValueHeaders']) and (event['multiValueHeaders']
['greeter']) and (
                event['multiValueHeaders']['greeter'] is not None):
            greeter = " and ".join(event['multiValueHeaders']['greeter'])
    except KeyError:
```

```
        print('No greeter')

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

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

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

    return res
```

8. Wählen Sie Deploy (Bereitstellen) aus.

Erstellen Sie eine "Hello, World!"- API

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

Erstellen von „Hello, World!“ API

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

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

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

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

Erstellen Sie eine Ressource

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

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

Erstellen einer **`ANY`**-Methode

1. Wählen Sie die /helloworld-Ressource aus und klicken Sie dann auf Methode erstellen.
2. Wählen Sie als Methodentyp die Option BELIEBIG aus.
3. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
4. Schalten Sie die Lambda-Proxy-Integration ein.

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

Bereitstellen und Testen der API

Stellen Sie Ihre API bereit

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

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

Sie können einen Browser oder eine cURL zum Testen Ihrer API verwenden.

Um `GET` Anfragen nur mit Abfragezeichenfolgenparametern zu testen, können Sie die URL für die `helloworld` API-Ressource in die Adressleiste eines Browsers eingeben.

Um die URL für die API-Ressource zu erstellen, fügen Sie die `helloworld` Ressource `helloworld` und den Abfragezeichenfolgenparameter `?greeter=John` an Ihre Aufruf-URL an. Ihre URL sollte wie folgt aussehen.

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

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

So testen Sie die bereitgestellte API mithilfe von cURL:

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

   > **Note**
   >
   > Wenn Sie den Befehl unter Windows ausführen, verwenden Sie stattdessen die folgende Syntax:
   >
   > ```
   > curl -v -X POST "https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/
   > test/helloworld" -H "content-type: application/json" -d "{ \"greeter\":
   >  \"John\" }"
   > ```

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

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

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

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

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

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

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

```
Hello, John!
```

Tutorial: API Gateway REST-API mit Lambda Nicht-Proxy-Integration erstellen

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

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

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

Themen

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

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

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

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

Node.js

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

console.log('Loading function');

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

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

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

  // Return a greeting to the caller
  callback(null, {
      "greeting": greeting
  });
};
```

Python

```
import json

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

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


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

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

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

Unsere Lambda-Funktion ist einfach. Sie analysiert das `event`-Objekt mit der Eingabe auf die Eigenschaften `name`, `city`, `time` und `day`. Sie gibt anschließend ein JSON-Objekt `{"message":greeting}` an den Aufrufer zurück. Die Nachricht nutzt das `"Good [morning|afternoon|day], [name|you] in [city|World]. Happy day!"`-Muster. Es wird angenommen, dass die Eingabe an die Lambda-Funktion das folgenden JSON-Objekt ist:

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

Weitere Informationen finden Sie im AWS Lambda -Entwicklerhandbuch.

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

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

Im Vergleich mit `GetStartedLambdaProxyIntegration`, der Lambda-Proxy-Integration, nimmt die `GetStartedLambdaIntegration` Lambda-Funktion für die benutzerdefinierte Lambda-Integration nur Eingaben aus den API Gateway API-Integrationsanforderungsdaten entgegen. Die Funktion kann eine Ausgabe als beliebiges JSON-Objekt, als Zeichenfolge, als Boolean oder sogar Binär-Blob zurückgeben. Die Lambda-Funktion für die Lambda-Proxy-Integration kann dagegen die Eingabe von beliebigen Anfragedaten entgegennehmen, muss aber ein bestimmtes JSON-Objekt

zurückgeben. Die `GetStartedLambdaIntegration`-Funktion für die benutzerdefinierte Lambda-Integration kann API-Anfrageparameter als Eingabe nutzen, sofern API Gateway die erforderlichen API-Anfrageparameter vor dem Weiterleiten der Anfrage an das Backend zum Anfragetext zuordnet. Hierzu muss der API-Entwickler beim Erstellen der API eine Mapping-Vorlage für die API-Methode erstellen und konfigurieren.

Erstellen Sie jetzt die `GetStartedLambdaIntegration` Lambda-Funktion.

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

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

b. Akzeptieren Sie die Standardwerte für alle anderen Felder in diesem Abschnitt.

c. Wählen Sie Deploy (Bereitstellen) aus.

6. Um die neu erstellte Funktion zu testen, wählen Sie die Registerkarte Test aus.

a. Geben Sie für Event name (Ereignisname) **`HelloWorldTest`** ein.

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

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

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

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

Die Ausgabe wird auch in CloudWatch Logs geschrieben.

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:region:account-id:*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:/aws/lambda/
GetStartedLambdaIntegration:*"
            ]
        }
    ]
}
```

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

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

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

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

API mit Lambda Nicht-Proxy-Integration erstellen

Wenn die Lambda-Funktion (`GetStartedLambdaIntegration`) erstellt und getestet ist, können Sie die Funktion über eine API Gateway-API zur Verfügung stellen. Zur Veranschaulichung

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

Eine API mit einer Lambda Nicht-Proxy-Integration erstellen

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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Geben Sie in API name (API-Name) **`LambdaNonProxyAPI`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie Create API (API erstellen) aus.

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

Erstellen Sie eine Ressource

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

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

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

Erstellen einer **ANY**-Methode

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

   Jetzt aktivieren Sie einen Anforderungsvalidator für eine URL-Pfadvariable, einen Abfragezeichenfolgenparameter und einen Header, um sicherzustellen, dass alle erforderlichen Daten definiert sind. Für dieses Beispiel erstellen Sie einen `time`-Abfragezeichenfolgenparameter und einen `day`-Header.
7. Wählen Sie für Anforderungs-Validierer die Option Abfragezeichenfolgeparameter und -Header validieren aus.
8. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:

   a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

   b. Geben Sie unter Name **`time`** ein.

   c. Aktivieren Sie die Option Erforderlich.

   d. Caching bleibt ausgeschaltet.
9. Klicken Sie auf HTTP-Anforderungs-Header und gehen Sie dann wie folgt vor:

   a. Wählen Sie Add header.

   b. Geben Sie unter Name **`day`** ein.

c. Aktivieren Sie die Option Erforderlich.

d. Caching bleibt ausgeschaltet.

10. Wählen Sie Methode erstellen aus.

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

Konfigurieren der Integrationsanforderung

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

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

7. Wählen Sie Save (Speichern) aus.

Testaufruf der API-Methode

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

```
POST /Seattle?time=morning
```

```
day:Wednesday

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

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

So testen Sie die **ANY**-Methode

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

Überprüfen Sie, ob die zurückgegebene Antwortnutzlast wie folgt lautet:

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

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

```
Execution log for request test-request
Thu Aug 31 01:07:25 UTC 2017 : Starting execution for request: test-invoke-request
Thu Aug 31 01:07:25 UTC 2017 : HTTP Method: POST, Resource Path: /Seattle
Thu Aug 31 01:07:25 UTC 2017 : Method request path: {city=Seattle}
Thu Aug 31 01:07:25 UTC 2017 : Method request query string: {time=morning}
Thu Aug 31 01:07:25 UTC 2017 : Method request headers: {day=Wednesday}
Thu Aug 31 01:07:25 UTC 2017 : Method request body before transformations:
 { "callerName": "John" }
Thu Aug 31 01:07:25 UTC 2017 : Request validation succeeded for content type
 application/json
```

```
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request URI: https://
lambda.us-west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-
tag=test-request,
 Authorization=**********************************************************************
 X-Amz-Date=20170831T010725Z, x-amzn-apigateway-api-id=beags1mnid, X-Amz-
Source-Arn=arn:aws:execute-api:us-west-2:123456789012:beags1mnid/null/POST/
{city}, Accept=application/json, User-Agent=AmazonAPIGateway_beags1mnid,
 X-Amz-Security-Token=FQoDYXdzELL//////////wEaDMHGzEdEOT/VvGhabiK3AzgKrJw
+3zLqJZG4PhOq12K6W21+QotY2rrZyOzqhLoiuRg3CAYNQ2eqgL5D54+63ey9bIdtwHGoyBdq8ecWxJK/
YUnT2Rau0L9HCG5p7FC05h3IvwlFfvcidQNXeYvsKJTLXI05/
yEnY3ttIAnpNYLOezD9Es8rBfyruHfJfOqextKlsC8DymCcqlGkig8qLKcZ0hWJWVwiPJiFgL7laabXs+
+ZhCa4hdZo4iqlG729DE4gaV1mJVdoAagIUwLMo+y4NxFDu0r7I0/
EO5nYcCrppGVVBYiGk7H4T6sXuhTkbNNqVmXtV3ch5bOlh7 [TRUNCATED]
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request body after transformations: {
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday",
  "name" : "John"
}
Thu Aug 31 01:07:25 UTC 2017 : Sending request to https://lambda.us-
west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Received response. Integration latency: 328 ms
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response body before transformations:
 {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response headers: {x-amzn-Remapped-Content-
Length=0, x-amzn-RequestId=c0475a28-8de8-11e7-8d3f-4183da788f0f, Connection=keep-
alive, Content-Length=62, Date=Thu, 31 Aug 2017 01:07:25 GMT, X-Amzn-Trace-
Id=root=1-59a7614d-373151b01b0713127e646635;sampled=0, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Method response body after transformations:
 {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampled=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200
```

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

Bereitstellen der API

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

So stellen Sie die API für eine Stufe bereit

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

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

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

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

Testen der API in einer Bereitstellungsstufe

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

So testen Sie die API mithilfe von cURL

1. Öffnen Sie ein Terminalfenster auf Ihrem lokalen Computer, der mit dem Internet verbunden ist.

2. So können Sie dies testen `POST /Seattle?time=evening`:

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

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

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

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

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

Bereinigen

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

> **Warning**
>
> Wenn Sie vorhaben, die anderen exemplarischen Vorgehensweisen in dieser Reihe abzuschließen, löschen Sie weder die Lambda-Ausführungsrolle noch die Lambda-Aufruf-Rolle. Wenn Sie eine Lambda-Funktion löschen, die von APIs benötigt wird, funktionieren diese APIs nicht mehr. Das Löschen einer Lambda-Funktion kann nicht rückgängig gemacht werden. Wenn Sie die Lambda-Funktion erneut verwenden möchten, müssen Sie die Funktion neu erstellen.
>
> Wenn Sie eine IAM-Ressource löschen, auf die eine Lambda-Funktion angewiesen ist, funktioniert diese Lambda-Funktion nicht mehr, und alle APIs, die auf dieser Funktion beruhen, funktionieren nicht mehr. Das Löschen einer IAM-Ressource kann nicht rückgängig gemacht werden. Wenn Sie die IAM-Ressource wieder nutzen möchten, müssen Sie diese neu erstellen.

So löschen Sie die Lambda-Funktion:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie GetStartedLambdaIntegrationin der Liste der Funktionen Aktionen und anschließend Funktion löschen aus. Wählen Sie bei Aufforderung erneut Löschen aus.

Löschen Sie die zugehörigen IAM-Ressourcen wie folgt:

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in Details Roles.
3. Wählen Sie aus der Rollenliste GetStartedLambdaIntegrationRoledie Option Rollenaktionen und anschließend Rolle löschen aus. Folgen Sie den Schritten in der Konsole, um die Rolle zu löschen.

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

Sie können jetzt eine AWS Lambda Funktion aus einem anderen AWS Konto als Backend für Ihre API-Integration verwenden. Jedes Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist. Dies erleichtert das zentrale Verwalten und Teilen von Lambda-Backend-Funktionen über verschiedene APIs hinweg.

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

API für die kontenübergreifende Lambda-Integration mit API Gateway erstellen

So erstellen Sie eine API

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

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

3. Geben Sie in API name (API-Name) **`CrossAccountLambdaAPI`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie Create API (API erstellen) aus.

Lambda-Integrationsfunktion in einem anderen Konto erstellen

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

Erstellen einer Lambda-Funktion in einem anderen Konto

1. Melden Sie sich bei der Lambda-Konsole in einem anderen Konto als dem an, in dem Sie Ihre API Gateway-API erstellt haben.
2. Wählen Sie Create function (Funktion erstellen).
3. Wählen Sie Author from scratch aus.
4. Verfahren Sie unter Author from scratch wie folgt:
   a. Geben Sie als Function name (Funktionsname) einen Namen ein.
   b. Wählen Sie aus der Dropdown-Liste Runtime (Laufzeit) eine unterstützte Node.js-Laufzeit aus.
   c. Erweitern Sie unter Permissions (Berechtigungen) den Bereich Choose or create an execution role (Ausführungsrolle wählen oder erstellen). Sie können eine Rolle erstellen oder eine vorhandene Rolle auswählen.
   d. Wählen Sie Create function aus, um fortzufahren.
5. Scrollen Sie nach unten bis zum Bereich Function code (Funktionscode).
6. Geben Sie die Node.js-Funktionsimplementierung aus the section called “Tutorial: Hello World-API mit Lambda-Proxy-Integration erstellen” ein.
7. Wählen Sie Deploy (Bereitstellen) aus.
8. Beachten Sie den vollständigen ARN für Ihre Funktion (in der rechten oberen Ecke des Bereichs der Lambda-Funktion). Sie benötigen diese, wenn Sie Ihre kontoübergreifende Lambda-Integration erstellen.

Kontoübergreifende Lambda-Integration konfigurieren

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

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

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

Erstellen Sie eine Ressource

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

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

So erstellen Sie eine **`GET`**-Methode

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

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

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

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

(Optional) Anzeige Ihrer aktualisierten Richtlinie

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

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

Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels

Sie können die Amazon API Gateway Gateway-Konsole verwenden, um eine einfache REST-API mit der HTTP-Integration für eine PetStore Website zu erstellen und zu testen. Die API-Definition ist als Swagger-2.0-Datei vorkonfiguriert. Nachdem Sie die API-Definition in API Gateway geladen haben, können Sie die API Gateway-Konsole verwenden, um die Grundstruktur der API zu untersuchen oder die API einfach bereitstellen und testen.

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

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

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

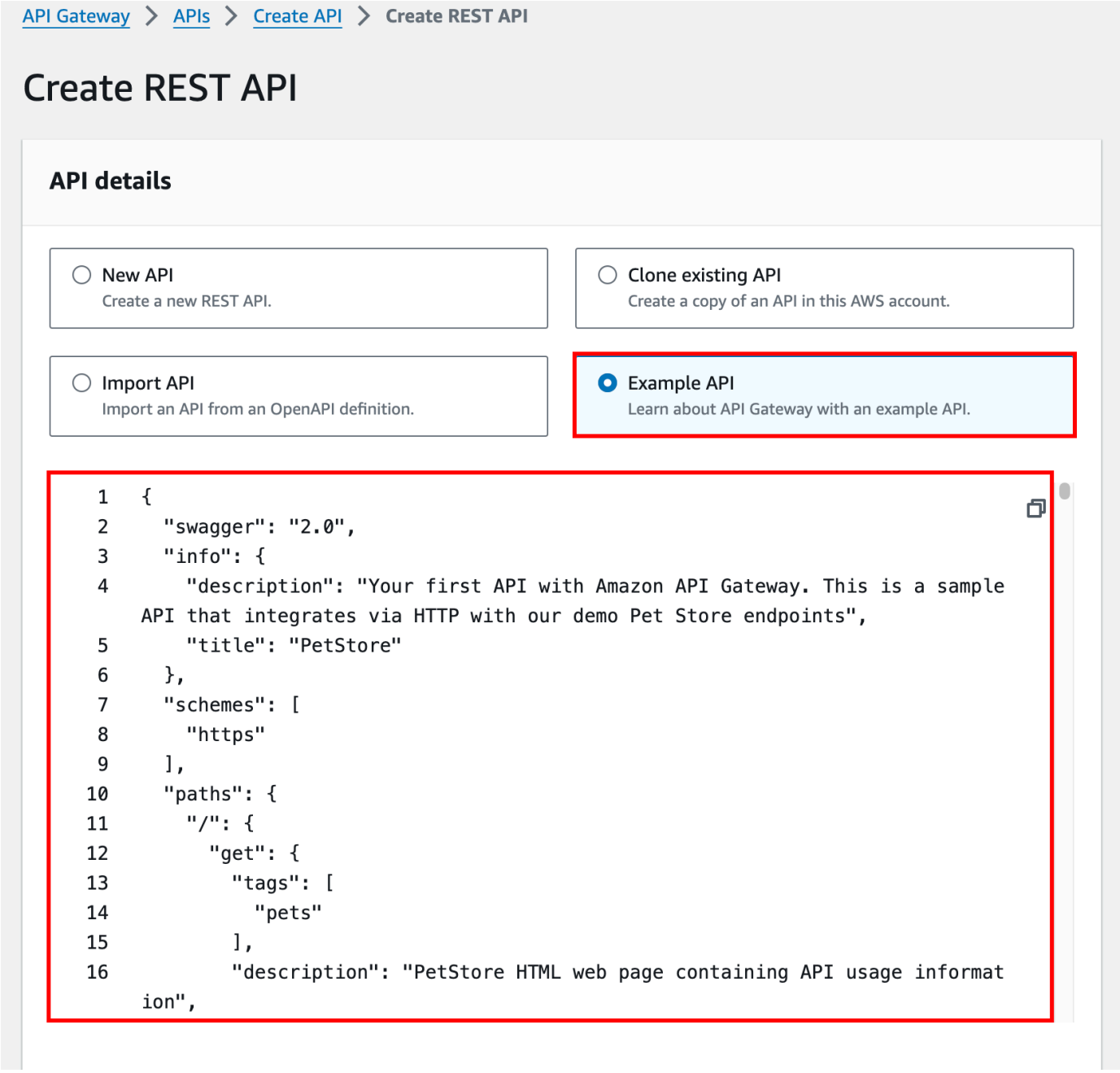
Die API unterstützt den CORS-Zugriff über die `OPTIONS`-Methoden des `MOCK`-Integrationstyps. API Gateway gibt die erforderlichen Header zurück, die den CORS-Zugriff unterstützen.

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

So importieren, erstellen und testen Sie die Beispiel-API

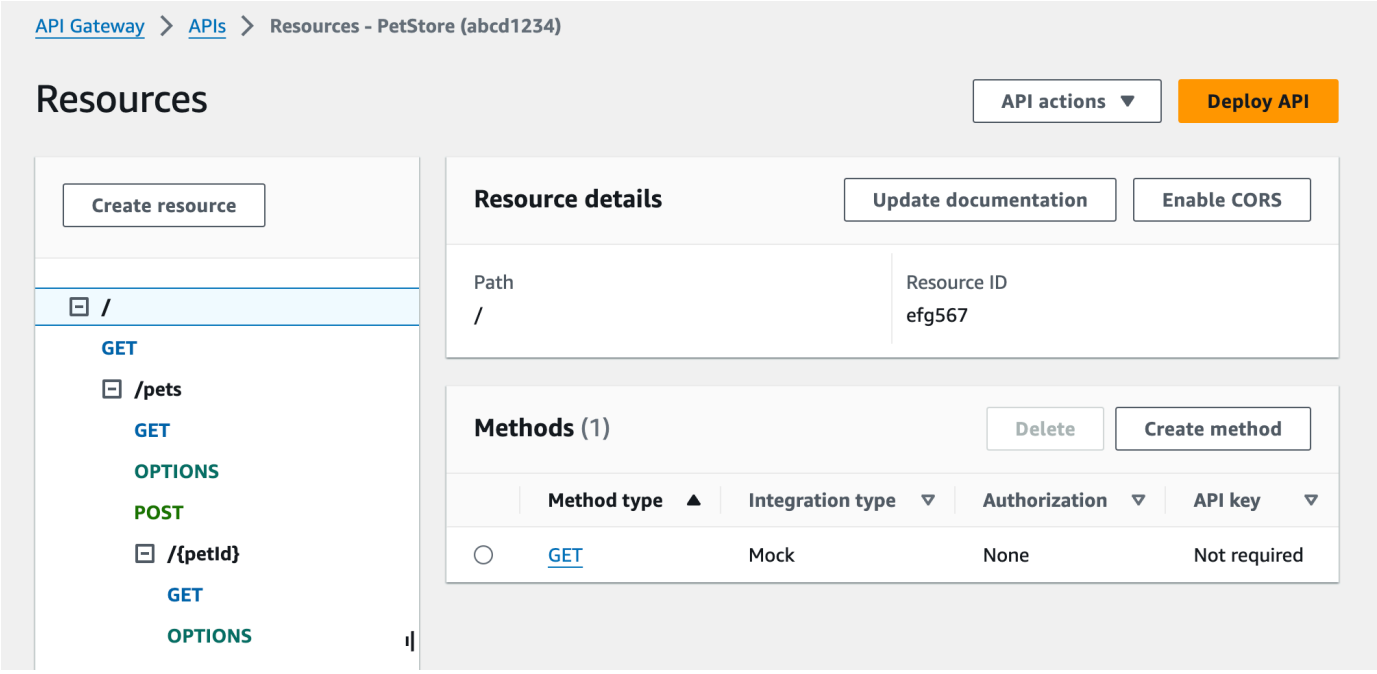
1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Führen Sie eine der folgenden Aktionen aus:
   - Um Ihre erste API zu erstellen, wählen Sie für REST-API die Option Erstellen.
   - Wenn Sie zuvor eine API erstellt haben, wählen Sie API erstellen und dann für REST-API die Option Erstellen.
3. Wählen Sie unter Create REST API (REST API erstellen) die Option Example API (Beispiel-API) und dann Create API (API erstellen) aus, um die Beispiel-API zu erstellen.



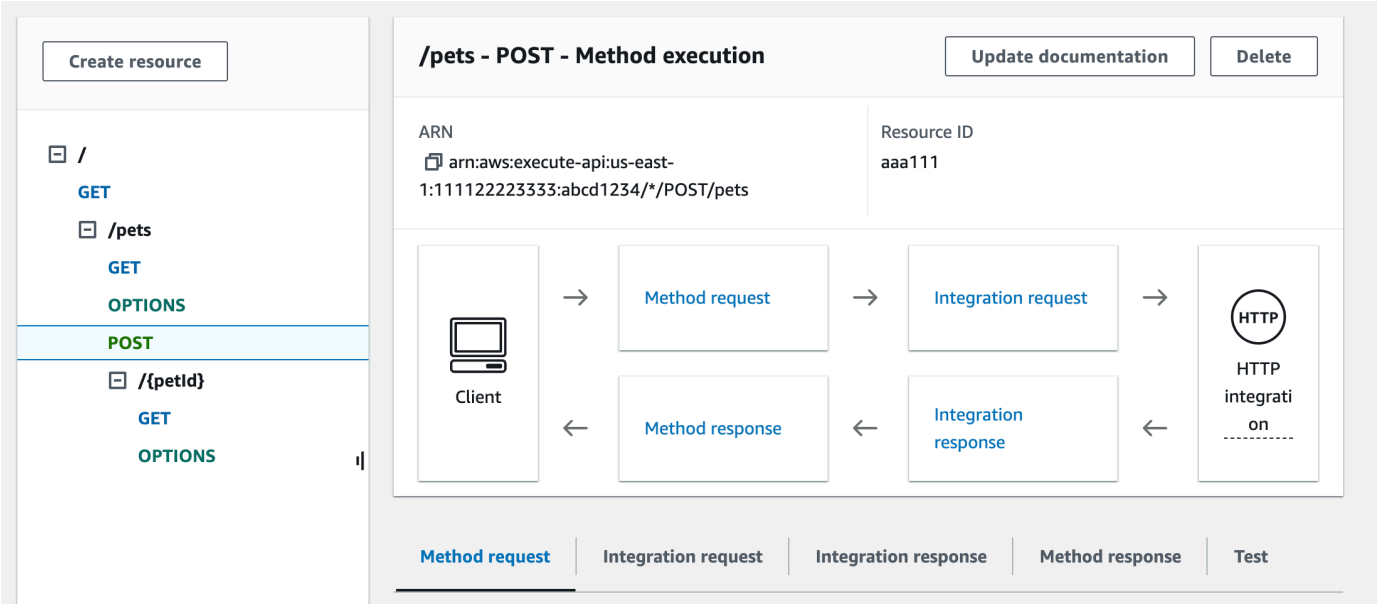
Sie können einen Bildlauf durch die OpenAPI-Definition ausführen und Details zu dieser Beispiel-API anzeigen. Klicken Sie anschließend auf Create API (API erstellen).

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



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

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



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

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

Ein Client verwendet die API für den Zugriff auf eine Backend-Funktion über Method Request (Methodenanforderung). Bei Bedarf wandelt Amazon API Gateway die Clientanforderung in ein Format um, das vom Backend in Integration Request (Integrationsanfrage) verarbeitet werden kann, bevor die eingehende Anfrage an das Backend weitergeleitet wird. Die umgewandelte Anforderung wird als Integrationsanforderung bezeichnet. Analog dazu gibt das Backend die Antwort an API Gateway in Integration Response (Integrationsantwort) zurück. API Gateway leitet dann an Method Response (Methodenantwort) weiter, bevor das Senden an den Client erfolgt. Auch hier kann API Gateway, falls erforderlich, die Backend-Antwortdaten einer vom Client erwarteten Form zuordnen.

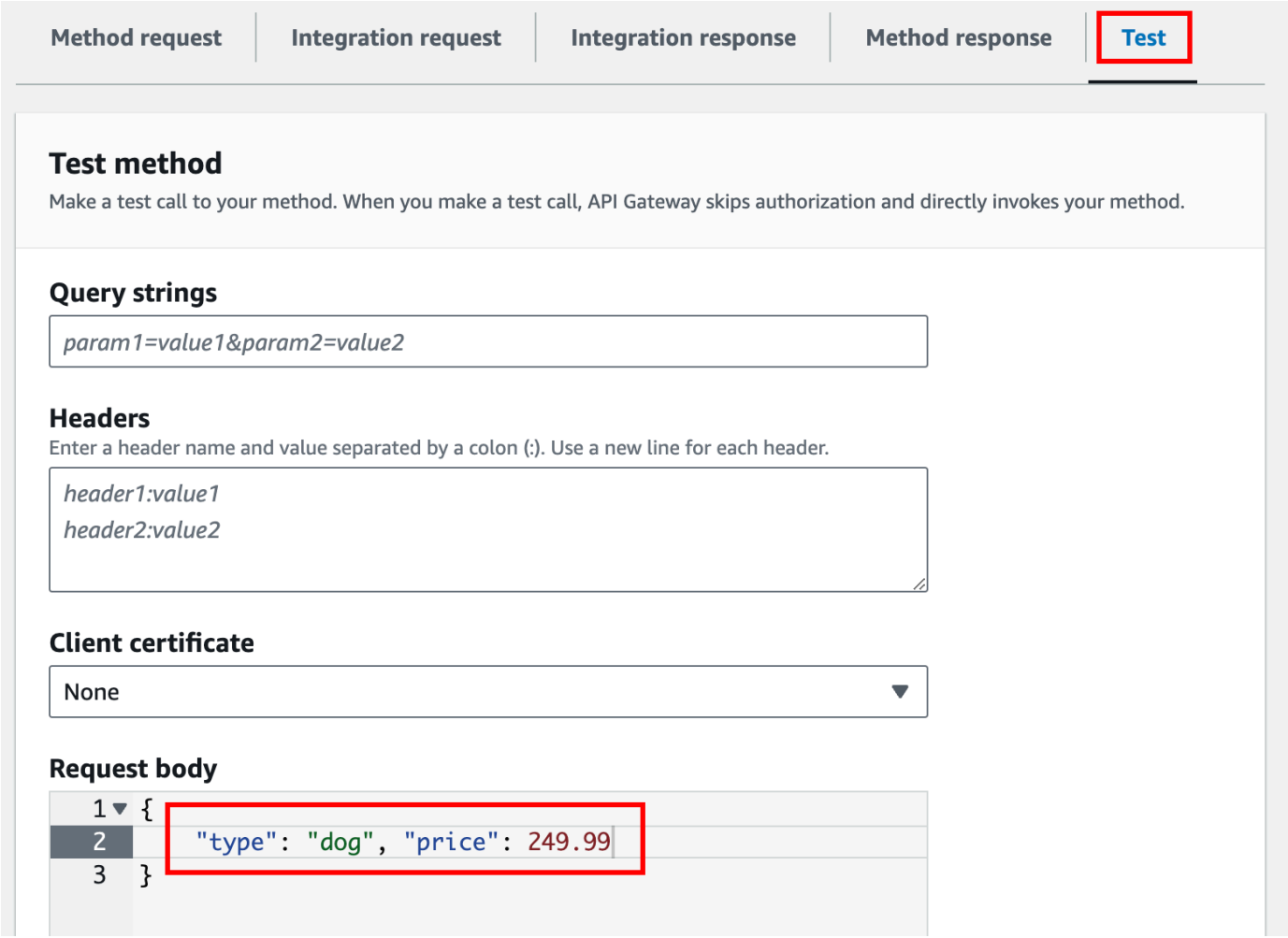
Für die `POST`-Methode einer API-Ressource kann die Nutzlast der Methodenanforderung ohne Änderung an die Integrationsanforderung übergeben werden, wenn die Nutzlast der Methodenanforderung dasselbe Format wie die Nutzlast der Integrationsanforderung aufweist.

Die `GET /`-Methodenanforderung verwendet den Integrationstyp `MOCK` und ist nicht mit einem tatsächlichen Endpunkt im Backend verknüpft. Die entsprechende Integration Response

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

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

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



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

8. Die Ergebnisse werden folgendermaßen angezeigt:

/pets - POST method test results

Request

/pets

Status

200

Latency

9

Response body


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


Response headers


```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65df8d2b-782cd3c572391cf4a85295f5"
}
```

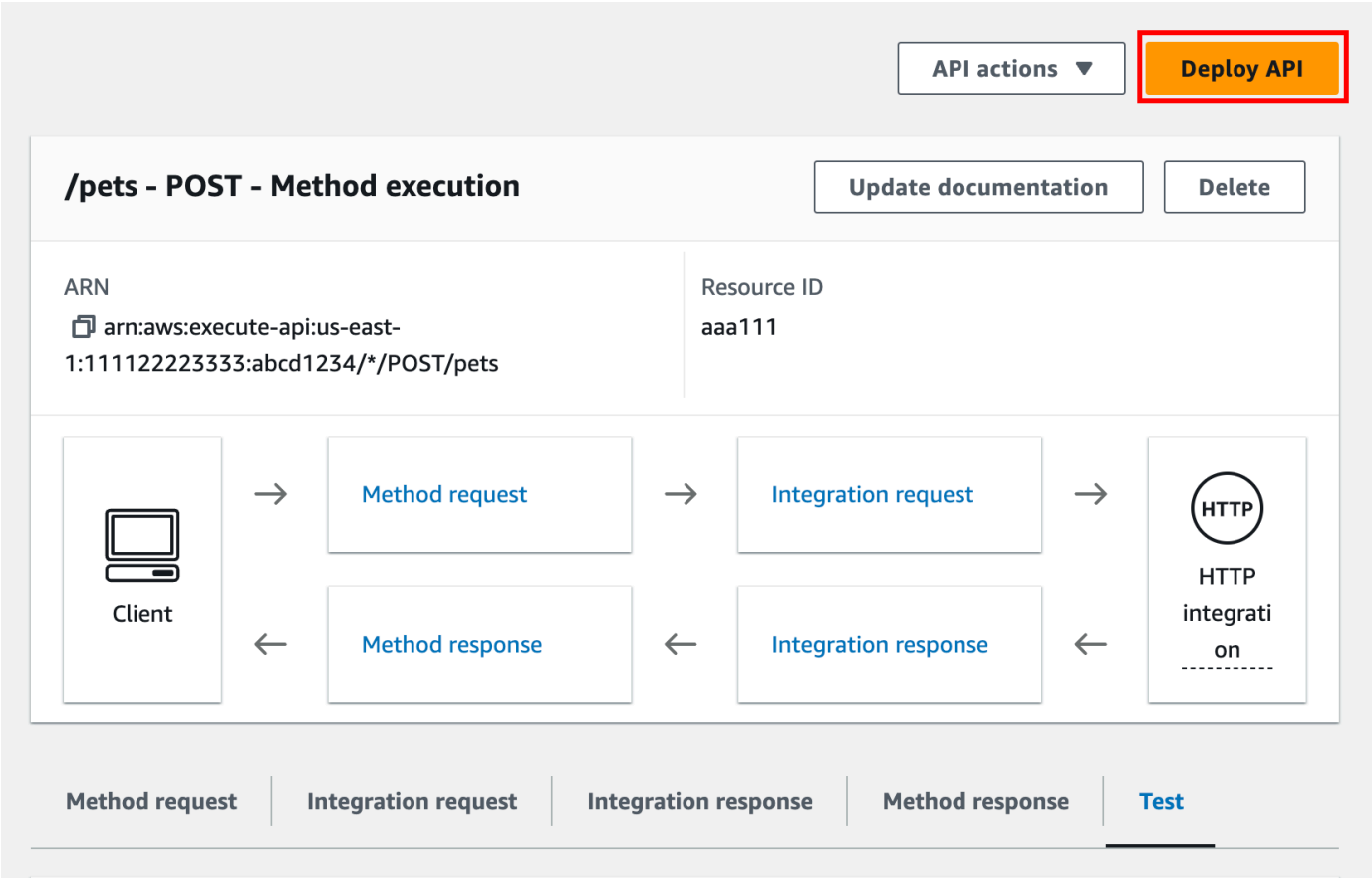

Log

```
Execution log for request 30f01060-307f-4447-803c-61679ea4c5d6
Wed Feb 28 19:44:43 UTC 2024 : Starting execution for request: 30f01060-
307f-4447-803c-61679ea4c5d6
```

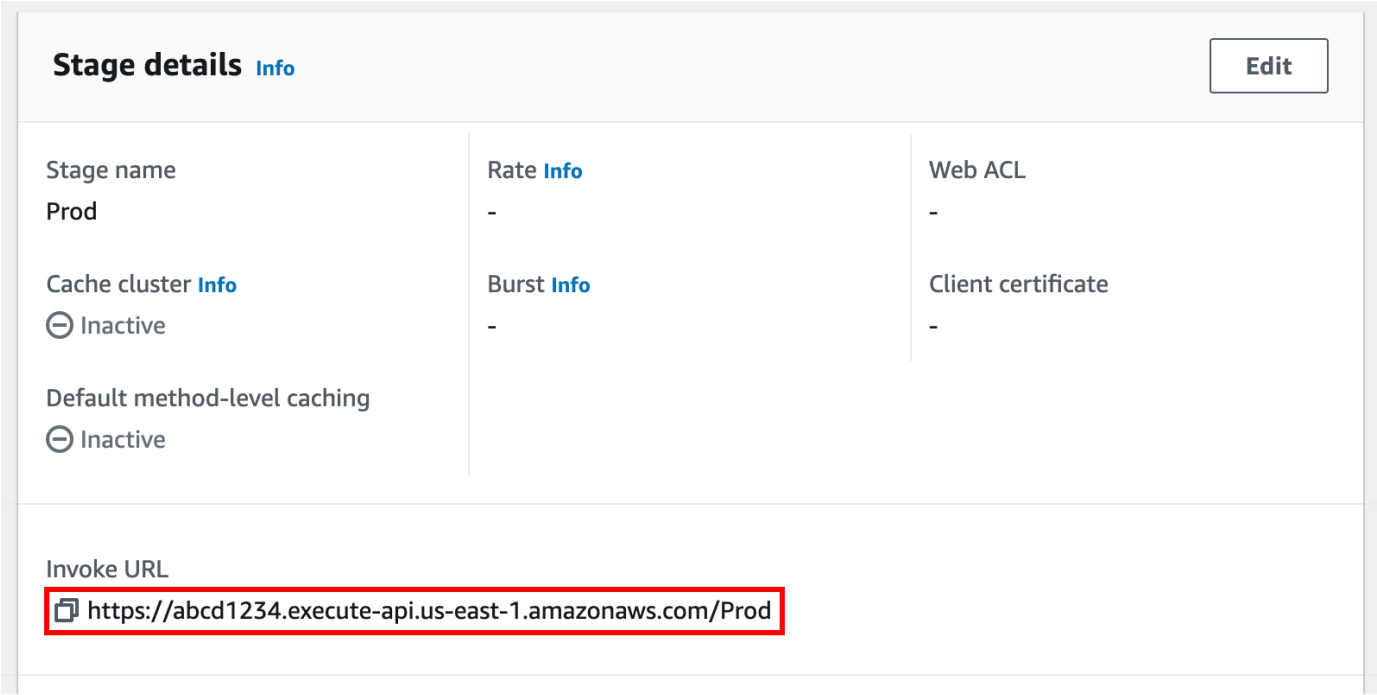
In der Ausgabe werden unter Log (Protokoll) die Statusänderungen von der Methodenanforderung zur Integrationsanforderung und von der Integrationsantwort zur Methodenantwort angezeigt. Das ist hilfreich, um Zuweisungsfehler zu beheben, die zu einer fehlerhaften Anforderung führen. In diesem Beispiel wird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

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

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



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



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

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

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

    Die API-Methode kann auf diese Weise aufgerufen werden, da der Authorization-Typ auf `NONE` festgelegt ist. Falls die `AWS_IAM`-Autorisierung verwendet wird, muss die Anfrage mit Signature Version 4 (SigV4)-Protokollen signiert werden. Ein Beispiel für eine solche Anforderung finden Sie unter the section called “Tutorial: Erstellen einer API mit Nicht-Proxy-HTTP-Integration”.

Wählen Sie ein Tutorial zur HTTP-Integration

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

Bei einer HTTP-Proxyintegration müssen Sie nur die HTTP-Methode und den HTTP-Endpunkt-URI entsprechend den Backend-Anforderungen festlegen. Wir empfehlen, dass Sie, wann immer möglich, die HTTP-Proxyintegration verwenden, um die Vorteile der optimierten API-Einrichtung zu nutzen.

Möglicherweise möchten Sie eine benutzerdefinierte HTTP-Integration verwenden, wenn Sie Kundenanforderungsdaten für das Backend oder die Backend-Antwortdaten für den Client transformieren müssen.

Themen

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

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

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

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

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

Themen

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

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

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

So erstellen Sie eine API

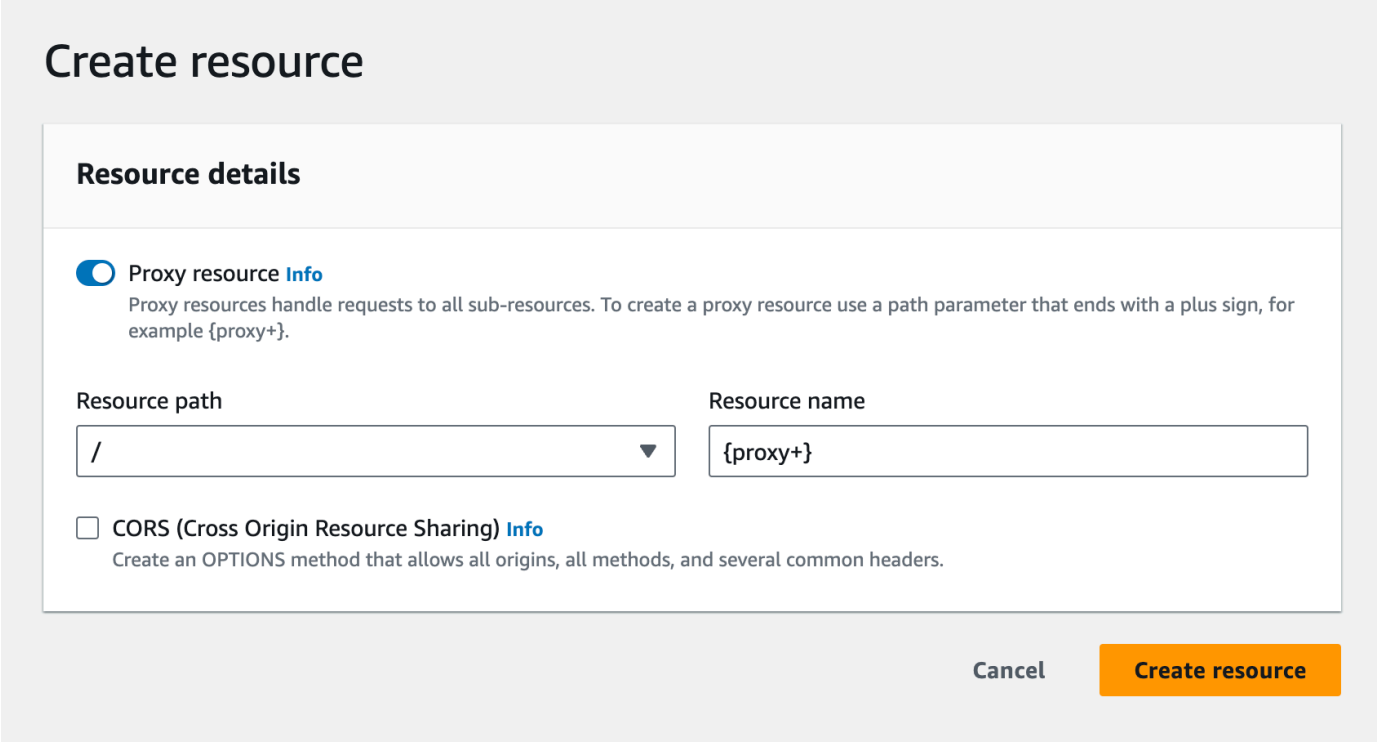
1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wenn Sie API Gateway zum ersten Mal verwenden, sehen Sie eine Seite, die Sie mit den Funktionen des Service vertraut macht. Wählen Sie unter REST-API die Option Erstellen aus. Wenn das Popup-Fenster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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Geben Sie in API name (API-Name) **`HTTPProxyAPI`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie Create API (API erstellen) aus.

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

So erstellen Sie eine /{proxy+}-Ressource

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



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

Erstellen einer **ANY**-Methode

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

Testen einer API mit HTTP-Proxy-Integration

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

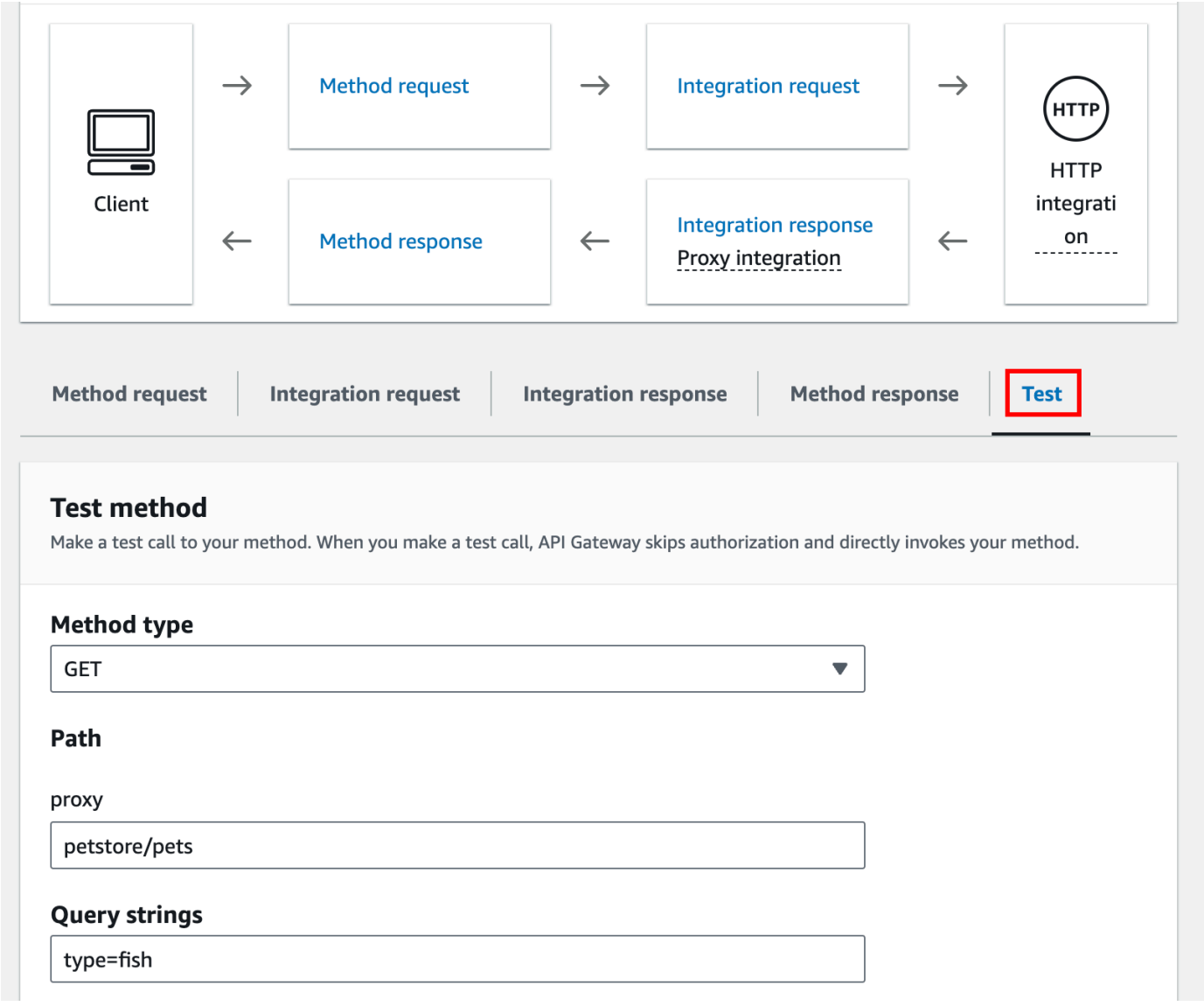
- Ob das Backend den entsprechenden Backend-Endpunkt verfügbar gemacht hat und, falls dies der Fall ist, ob die erforderlichen Zugriffsberechtigungen erteilt wurden.
- Ob der Client die korrekte Eingabe bereitstellt.

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

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

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

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



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

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

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

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

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

Stellen Sie Ihre API bereit

1. Klicken Sie auf Deploy API.

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

Jetzt können Clients Ihre API aufrufen.

So rufen Sie Ihre API auf

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

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

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

   Ihr Browser sendet eine `GET`-Anforderung an die API.
5. Das Ergebnis sollte mit dem übereinstimmen, was Ihr Test in der API Gateway-Konsole zurückgegeben hat.

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

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

Themen

- Erstellen einer API mit benutzerdefinierter HTTP-Integration
- (Optionale) Parameter für die Zuordnungsanforderung

Erstellen einer API mit benutzerdefinierter HTTP-Integration

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

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

So erstellen Sie eine API

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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Geben Sie in API name (API-Name) **`HTTPNonProxyAPI`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie Create API (API erstellen) aus.

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

Erstellen einer /pets-Ressource

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

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

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

- POST; wird vor allem für die Erstellung von untergeordneten Ressourcen verwendet.
- PUT; wird vor allem für die Aktualisierung vorhandener Ressourcen verwendet (kann auch für die Erstellung untergeordneter Ressourcen verwendet werden, obwohl dies nicht empfohlen wird).
- DELETE; wird für das Löschen von Ressourcen verwendet.
- PATCH; wird für die Aktualisierung von Ressourcen verwendet.
- HEAD; wird vor allem in Testszenarien verwendet. Diese Option ist identisch mit GET, gibt aber nicht die Repräsentation der Ressource zurück.
- OPTIONS – kann von Aufrufern verwendet werden, um Informationen über verfügbare Kommunikationsoptionen für den Zielservice abzurufen.

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

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

1. Wählen Sie die /pets-Ressource aus.
2. Wählen Sie Methode erstellen aus.
3. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
4. Für den Integrationstyp wählen Sie die Option HTTP-Integration aus.
5. Die Option Lambda-Proxy-Integration bleibt deaktiviert.

6. Für HTTP-Methode wählen Sie GET aus.
7. Geben Sie für Endpunkt-URL **`http://petstore-demo-endpoint.execute-api.com/petstore/pets`** ein.

   Auf der PetStore Website können Sie auf einer bestimmten Seite eine Liste von `Pet` Objekten abrufen, die nach der Art des Haustiers, z. B. „Hund“ oder „Katze“, sortiert sind.
8. Für Inhaltsbehandlung wählen Sie Passthrough aus.
9. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

   Die PetStore Website verwendet die Parameter `type` und die `page` Abfragezeichenfolge, um eine Eingabe zu akzeptieren. Sie fügen der Methodenanforderung Abfragezeichenfolgenparameter hinzu und ordnen sie den entsprechenden Abfragezeichenfolgenparametern der Integrationsanforderung zu.
10. Gehen Sie wie folgt vor, um die Abfragezeichenfolgenparameter hinzuzufügen:

    a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

    b. Für Name geben Sie **`type`** ein.

    c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.

    Wiederholen Sie die obigen Schritte, um eine zusätzliche Abfragezeichenfolge namens **`page`** zu erstellen.
11. Wählen Sie Methode erstellen aus.

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

Zuordnen von Eingabeparametern zur Integrationsanforderung

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Abfragezeichenfolgen-Parameter und gehen Sie dann wie folgt vor:

   a. Klicken Sie auf Abfragezeichenfolge-Parameter hinzufügen.

   b. Geben Sie unter Name **`type`** ein.

c. Für Zugeordnet von geben Sie **`method.request.querystring.type`** ein.

d. Caching bleibt ausgeschaltet.

e. Klicken Sie auf Abfragezeichenfolge-Parameter hinzufügen.

f. Geben Sie unter Name **`page`** ein.

g. Für Zugeordnet von geben Sie **`method.request.querystring.page`** ein.

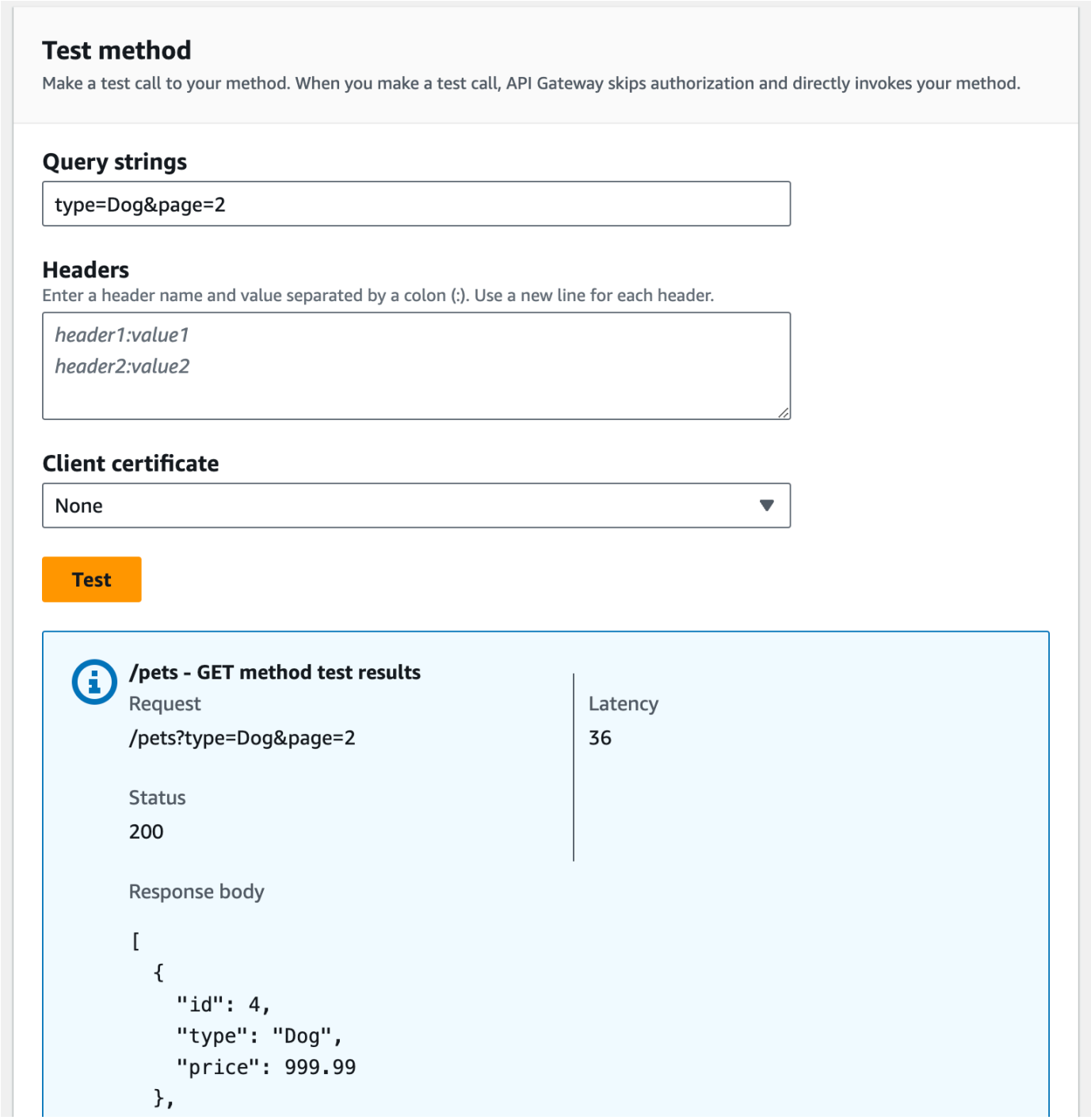
h. Caching bleibt ausgeschaltet.

3. Wählen Sie Speichern.

So testen Sie die API-Daten

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

   Das Ergebnis sollte wie folgt aussehen:



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

4. Klicken Sie auf Deploy API.
5. Wählen Sie für Stufe die Option Neue Stufe aus.
6. Geben Sie für Stage name (Stufenname) **`Prod`** ein.

7. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
8. Wählen Sie Bereitstellen.
9. (Optional) Klicken Sie unter Stufendetails auf das Kopiersymbol für Aufruf-URL, um die URL Ihrer Aufruf-API zu kopieren. Sie können diese mit Tools wie Postman und cURL zum Testen Ihrer API verwenden.

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

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

(Optionale) Parameter für die Zuordnungsanforderung

Anfrageparameter für ein API Gateway API zuordnen

In diesem Tutorium erfahren Sie, wie sie einen `{petId}`-Pfadparameter über die Methodenanforderungs-URL der API erstellen, um eine Element-ID anzugeben, diese dem `{id}`-Pfadparameter in der Integrationsanforderungs-URL zuzuweisen, und dann die Anforderung an den HTTP-Endpunkt zu senden.

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

Schritt 1: Erstellen von Ressourcen

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

Erstellen der {peTid}-Ressource

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

2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Für Ressourcenpfad wählen Sie die Option/pets/aus.
4. Geben Sie für Resource name (Ressourcenname) **`{petId}`** ein.

   Setzen Sie geschweifte Klammern (`{ }`) um `petId`, sodass /pets/{petId} angezeigt wird.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.

Schritt 2: Erstellen und Testen der Methoden

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

Einrichten der GET-Methode

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

Als Nächstes ordnen Sie den `{petId}`-Pfadparameter dem `{id}`-Pfadparameter im HTTP-Endpunkt zu.

Zuordnen des **`{petId}`**-Pfadparameters

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Pfadparameter.

3. API Gateway erstellt einen Pfadparameter für die Integrationsanforderung mit dem Namen petId. Das funktioniert nicht für dein Backend. Der HTTP-Endpunkt verwendet `{id}` als Pfadparameter. Sie müssen petId in **`id`** umbenennen.

   Damit wird der Pfadparameter `petId` der Methodenanforderung dem Pfadparameter `id` der Integrationsanforderung zugewiesen.
4. Wählen Sie Speichern.

Jetzt können Sie die Methode testen.

So testen Sie die -Methode

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

   Wenn diese Aktion erfolgreich ist, zeigt Antworttext Folgendes an:

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

Schritt 3: Bereitstellen der API

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

So stellen Sie die API bereit

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

Schritt 4: Testen der API

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

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

   Sie sollte wie folgt aussehen:

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

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

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

Nächste Schritte

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

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

- Weitere Informationen zur Anforderungsvalidierung finden Sie unter Grundlegende Anforderungsvalidierung in API Gateway einrichten.
- Weitere Informationen zur Umwandlung von Anforderungs- und Antwort-Nutzlasten finden Sie unter Einrichten von Datentransformationen in API Gateway.
- Informationen zum Erstellen benutzerdefinierter Gateway-Antworten finden Sie unter Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole.

Tutorial: REST-API mit privater API Gateway-Integration erstellen

Sie können eine API Gateway-API mit einer privaten Integration erstellen, um Ihren Kunden Zugang zu HTTP/HTTPS-Ressourcen innerhalb Ihrer Amazon Virtual Private Cloud (Amazon VPC) zu bieten. Bei diesen VPC-Ressourcen handelt es sich um HTTP-/HTTPS-Endpunkte auf einer EC2-Instance in einem Network Load Balancer in der VPC. Der Network Load Balancer kapselt die VPC-Ressourcen und leitet eingehende Anfragen zur Zielressource weiter.

Wenn ein Kunde die API aufruft, verbindet sich API Gateway über die vorkonfigurierte VPC-Verbindung mit dem Network Load Balancer. Ein VPC-Link ist durch eine API-Gateway-Ressource von gekapselt. VpcLink Er ist zuständig für die Weiterleitung der API-Methodenanforderungen an die VPC-Ressourcen und gibt Backend-Antworten an den Aufrufer zurück. Für einen API-Entwickler ist ein `VpcLink` funktional mit einem Integrationsendpunkt gleichwertig.

Wenn Sie eine API mit privater Integration erstellen möchten, müssen Sie einen neuen `VpcLink` erstellen oder einen vorhandenen auswählen, der mit einem Network Load Balancer verbunden ist, dessen Ziel die gewünschten VPC-Ressourcen sind. Sie benötigen entsprechende Berechtigungen, um einen `VpcLink` zu erstellen und zu verwalten. Anschließend richten Sie eine API--Methode ein und integrieren sie im `VpcLink`, indem Sie entweder `HTTP` oder `HTTP_PROXY` als Integrationstyp und `VPC_LINK` als Integrations-Verbindungstyp und die `VpcLink`-ID für die Integration `connectionId` festlegen.

> **Note**
>
> Der Network Load Balancer und die API müssen demselben AWS Konto gehören.

Damit Sie schnell eine API erstellen können, um Zugriff auf VPC-Ressourcen zu erhalten, gehen wir die wichtigsten Schritte zur Erstellung einer API mit privater Integration über die API Gateway-Konsole durch. Führen Sie vor dem Erstellen der API die folgenden Schritte aus:

1. Erstellen Sie eine VPC-Ressource, erstellen Sie in Ihrem Konto in der gleichen Region einen Network Load Balancer oder wählen Sie einen aus und fügen Sie die EC2-Instance, die die Ressource hostet, als Ziel des Network Load Balancers hinzu. Weitere Informationen finden Sie unter Network Load Balancer für private API Gateway-Integrationen einrichten.
2. Erteilen Sie Berechtigungen zum Erstellen des VPC-Links für private Integrationen. Weitere Informationen finden Sie unter Erteilen von Berechtigungen zum Erstellen eines VPC-Links.

Nachdem Sie Ihre VPC-Ressource erstellt und Ihren Network Load Balancer mit Ihrer VPC-Ressource in dessen Zielgruppen konfiguriert haben, führen Sie die folgenden Anweisungen aus, um eine API zu erstellen und über einen `VpcLink` in einer privaten Integration in der VPC-Ressource zu integrieren.

Eine API mit privater Integration erstellen

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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Sie können eine Edge-optimierte oder regionale REST-API erstellen.
4. Wählen Sie Ihre API aus.
5. Wählen Sie Methode erstellen aus und gehen Sie dann wie folgt vor:

   a. Wählen Sie als Methodentyp die Option `GET` aus.

   b. Für Integrationstyp wählen Sie VPC-Link aus.

   c. Aktivieren Sie die Lambda-Proxy-Integration.

   d. Für HTTP-Methode wählen Sie `GET` aus.

   e. Wählen Sie für VPC-Link [Stufenvariable verwenden] aus und geben Sie **`${stageVariables.vpcLinkId}`** in das Textfeld unten ein.

      Sie definieren die `vpcLinkId`-Stufenvariable nach der Bereitstellung der API für eine Stufe und setzen den Wert auf die ID des `VpcLink`.

   f. Geben Sie für Endpunkt-URL eine URL ein, zum Beispiel `http://myApi.example.com`.

      Hier wird der Hostnamen (z. B. `myApi.example.com`) verwendet, um den `Host`-Header der Integrationsanforderung festzulegen.

   g. Wählen Sie Methode erstellen aus.

      Mit der Proxy-Integration kann die API bereitgestellt werden. Andernfalls müssen Sie entsprechende Methoden- und Integrationsantworten einrichten.

6. Wählen Sie API bereitstellen aus und gehen Sie dann wie folgt vor:

   a. Wählen Sie für Stufe die Option Neue Stufe aus.

   b. Geben Sie unter Stage name (Stufenname) einen Namen für die Stufe ein.

   c. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.

   d. Wählen Sie Bereitstellen.

7. Notieren Sie sich die resultierende Aufruf-URL im Abschnitt Stufendetails. Diese ist zum Aufruf der API erforderlich. Davor müssen Sie die `vpcLinkId`-Stufenvariablen einrichten.

8. Wählen Sie im Bereich Stufen die Registerkarte Stufenvariablen aus und gehen Sie dann wie folgt vor:

   a. Wählen Sie Variablen verwalten aus und klicken Sie dann auf Stufenvariable hinzufügen.

   b. Geben Sie unter Name **`vpcLinkId`** ein.

   c. Geben Sie unter Wert die ID von`VPC_LINK` ein, zum Beispiel *`gix6s7`*.

   d. Wählen Sie Speichern.

      Mit der Stufenvariable können Sie ganz einfach auf verschiedene VPC-Links für die API wechseln, indem Sie den Wert der Stufenvariablen ändern.

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

In Tutorial: Hello World REST-API mit Lambda-Proxy-Integration erstellen und Wählen Sie ein AWS Lambda Integrations-Tutorial wird beschrieben, wie eine API Gateway-API erstellt wird, um die integrierte Lambda-Funktion zu bereitzustellen. Darüber hinaus können Sie eine API-Gateway-API erstellen, um andere AWS Dienste wie Amazon SNS, Amazon S3, Amazon Kinesis usw. verfügbar zu machen. AWS Lambda Dies wird durch die `AWS`-Integration ermöglicht. Die Lambda-Integration oder die Lambda-Proxy-Integration ist ein Sonderfall, bei dem der Aufruf der Lambda-Funktion über die API Gateway-API bereitgestellt wird.

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

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

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

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

Diese schrittweise Anleitung basiert auf den Anweisungen und Konzepten in Wählen Sie ein AWS Lambda Integrations-Tutorial. Wenn Sie diese Anleitung noch nicht ausgeführt haben, sollen Sie dies zuerst nachholen.

Themen

Voraussetzungen

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

1. Führen Sie die Schritte unter au Voraussetzungen für die ersten Schritte mit API Gateway.
2. Erstellen Sie eine neue API mit dem Namen `MyDemoAPI`. Weitere Informationen finden Sie unter Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen.

3. Stellen Sie die API mindestens einmal für eine Stufe mit dem Namen berei `test`. Weitere Informationen finden Sie unter Bereitstellen der API in Wählen Sie ein AWS Lambda Integrations-Tutorial.
4. Führen Sie die restlichen Schritte in au Wählen Sie ein AWS Lambda Integrations-Tutorial.
5. Erstellen Sie mindestens ein Thema in Amazon Simple Notification Service (Amazon SNS). Sie verwenden die bereitgestellte API, um eine Liste der Themen in Amazon SNS abzurufen, die mit Ihrem AWS Konto verknüpft sind. Wie Sie ein Thema in Amazon SNS erstellen, erfahren Sie unter Thema erstellen. (Sie müssen nicht die in Schritt 5 erwähnte Thema-ARN kopieren.)

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

Einer IAM-Rolle müssen geeignete IAM-Richtlinien zugeordnet sein, damit die API Amazon-SNS-Aktionen aufrufen kann.

Um die AWS Service-Proxy-Ausführungsrolle zu erstellen

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

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

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

Schritt 2: Erstellen der Ressource

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

So erstellen Sie die Ressource

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

Schritt 3: Erstellen der GET-Methode

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

So erstellen Sie die **`GET`**-Methode

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

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

Schritt 4: Angeben von Methodeneinstellungen und Testen der Methode

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

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

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

   Im Ergebnis wirde eine Antwort ähnlich der Folgenden angezeigt:

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

```
        }
      ]
    },
    "ResponseMetadata": {
      "RequestId": "abc1de23-45fa-6789-b0c1-d2e345fa6b78"
    }
  }
}
```

Schritt 5: Bereitstellen der API

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

So stellen Sie die API bereit

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

Schritt 6: Testen der API

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

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

   Das sollte wie folgt aussehen:

   ```
   https://my-api-id.execute-api.region-id.amazonaws.com/test
   ```
3. Geben Sie die URL in das Adressfeld einer neuen Registerkarte im Browser ein.
4. Fügen Sie `/mydemoawsproxy` an, sodass die URL wie folgt aussieht:

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

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

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

Schritt 7: Bereinigen

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

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

Löschen Sie die zugehörigen IAM-Ressourcen wie folgt:

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Bereich Details die Option Roles aus.
3. Wählen Sie ApiGateway und dann RollenaktionenAWSProxyExecRole, Rolle löschen aus. Wählen Sie bei Aufforderung Yes, Delete.
4. Wählen Sie im Bereich Details Policies.
5. Wählen Sie ApiGateway und dann Policy ActionsAWSProxyExecPolicy, Delete aus. Wählen Sie bei Aufforderung Löschen.

Sie haben das Ende dieser Anleitung erreicht. Ausführlichere Erläuterungen zum Erstellen einer API als AWS Dienstproxy finden Sie unter, Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API GatewayTutorial: Erstellen Sie eine Rechner-REST-API mit zwei AWS Serviceintegrationen und einer Lambda-Non-Proxy-Integration, oder. Tutorial: REST-API als Amazon Kinesis-Proxy in API Gateway erstellen

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

Im Erste-Schritte-Tutorial zur Nicht-Proxy-Integration wird ausschließlich die `Lambda Function`-Integration verwendet. Die `Lambda Function`-Integration ist ein Sonderfall des `AWS Service`-Integrationstyps, der einen Großteil der Integrationseinrichtung für Sie durchführt, z. B. das automatische Hinzufügen der erforderlichen ressourcenbasierten Berechtigungen zum Aufrufen der Lambda-Funktion. Hier wird bei zwei der drei Integrationen die `AWS Service`-Integration verwendet. Bei diesem Integrationstyp haben Sie mehr Kontrolle, aber Sie müssen bestimmte Aufgaben manuell ausführen. Hierzu gehört z. B. das Erstellen und Angeben einer IAM-Rolle mit entsprechenden Berechtigungen.

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

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

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

Themen

- Übernehmbare IAM-Rolle erstellen
- Erstellen einer Calc-Lambda-Funktion
- Calc Lambda-Funktion testen
- Erstellen einer Calc-API

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

Übernehmbare IAM-Rolle erstellen

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

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

Die Rolle, die Sie erstellen, benötigt eine InvokeFunctionLambda-Berechtigung. Andernfalls erhält der API-Aufrufer eine `500 Internal Server Error`-Antwort. Um der Rolle diese Berechtigung zu erteilen, hängen ihr die folgende IAM-Richtlinie an:

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

```
        }
    ]
}
```

Gehen Sie hierzu folgendermaßen vor:

API Gateway erstellen, das die IAM-Rolle übernimmt

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

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

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

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

   b. Wählen Sie Review policy (Richtlinie prüfen) aus.

   c. Gehen Sie unter Review Policy wie folgt vor:

      i. Geben Sie unter Name einen Namen wie **`lambda_execute`** ein.

ii. Wählen Sie Create Policy (Richtlinie erstellen) aus.

8. Führen Sie im ursprünglichen Konsolen-Fenster Create Role Folgendes aus:

   a. Wählen Sie unter Attach permissions policies (Berechtigungsrichtlinien hinzufügen), wählen die **`lambda_execute`** Richtlinie aus der Dropdown-Liste aus.

      Wenn Ihre Richtlinie nicht in der Liste angezeigt wird, wählen Sie die Schaltfläche "Aktualisieren" oben in der Liste aus. (Aktualisieren Sie nicht die Browserseite!)

   b. Wählen Sie Next: Tags (Weiter: Tags) aus.

   c. Klicken Sie auf Weiter: Prüfen.

   d. Geben Sie unter Role name einen Namen wie **`lambda_invoke_function_assume_apigw_role`** ein.

   e. Wählen Sie Create role aus.

9. Wählen Sie Ihre **`lambda_invoke_function_assume_apigw_role`** aus der Rollen-Liste aus.

10. Wählen Sie die Registerkarte Trust relationships (Vertrauensstellungen).

11. Wählen Sie Edit Trust Relationship (Vertrauensstellungen bearbeiten).

12. Ersetzen Sie die vorhandene Richtlinie durch Folgendes:

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

13. Wählen Sie Update Trust Policy.

14. Notieren Sie sich die Rolle ARN für die Rolle, die Sie gerade erstellt haben. Sie benötigen sie zu einem späteren Zeitpunkt.

Erstellen einer **`Calc`**-Lambda-Funktion

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

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

   Node.js

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

  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
    return "400 Invalid Input";
  }

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
    case "add":
      res.c = res.a + res.b;
      break;
```

```
    case "-":
    case "sub":
      res.c = res.a - res.b;
      break;
    case "*":
    case "mul":
      res.c = res.a * res.b;
      break;
    case "/":
    case "div":
      if (res.b == 0) {
        return "400 Divide by Zero";
      } else {
        res.c = res.a / res.b;
      }
      break;
    default:
      return "400 Invalid Operator";
  }

  return res;
};
```

Python

```
import json


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

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

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

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

    return res
```

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

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

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

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

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

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

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

Calc Lambda-Funktion testen

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

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

   ```
   {
     "a": "2",
     "b": "5",
     "op": "+"
   }
   ```
4. Wählen Sie Save (Speichern) aus.
5. Wählen Sie Test aus.
6. Erweitern Sie Execution result: succeeded (Ausführungsergebnis: erfolgreich). Sie sollten Folgendes sehen:

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

Erstellen einer **Calc**-API

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

So erstellen Sie eine API

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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Geben Sie in API name (API-Name) **`LambdaCalc`** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Lassen Sie die Einstellung für API-Endpunkttyp bei Regional.
6. Wählen Sie Create API (API erstellen) aus.

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

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

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

So erstellen Sie eine /calc-Ressource

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

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

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

So erstellen Sie eine **GET**-Methode

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

    Mit dieser Einstellung wird eine Fehlermeldung zurückgegeben, falls der Client die erforderlichen Parameter nicht vorgibt.
14. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

Jetzt richten Sie Abfragezeichenfolgenparameter für die GET-Methode in der Ressource /calc ein, sodass sie Eingaben im Namen der Backend-Lambda-Funktion empfangen kann.

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

a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

b. Geben Sie unter Name **`operand1`** ein.

c. Aktivieren Sie die Option Erforderlich.

d. Caching bleibt ausgeschaltet.

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

15. Wählen Sie Methode erstellen aus.

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

Zuordnen von Eingabeparametern zur Integrationsanforderung

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

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

```
    "op": "$input.params('operator')"
}
```

7. Wählen Sie Speichern.

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

So testen Sie die **GET**-Methode

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

   Die Ergebnisse sollten in etwa wie folgt aussehen:

Test method

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

Query strings

operand1=2&operand2=3&operator=+

Headers

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

header1:value1
header2:value2

Client certificate

None

Test

/ - GET method test results

Request

/?
operand1=2&operand2=3&operator=+

Status

200

Latency

414

Response body

```
{"a":2,"b":3,"op":"+","c":5}
```

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

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

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

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

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

Erstellen eines Eingabemodells

1. Klicken Sie im Navigationsbereich auf Models (Modelle).

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

   Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **`$default`** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
5. Geben Sie für Modellschema Folgendes ein:

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

6. Wählen Sie Modell erstellen aus.

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

Erstellen eines Ausgabemodells

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

   Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **`$default`** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
4. Geben Sie für Modellschema Folgendes ein:

```
{
```

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

5. Wählen Sie Modell erstellen aus.

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

Erstellen eines Ergebnismodells

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

   Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **`$default`** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
4. Geben Sie für Modellschema das folgende Modell mit Ihrer *`Restapi-ID`* ein. Ihre *`Restapi-ID`* wird im folgenden Flow oben in der Konsole in Klammern aufgeführt: `API Gateway > APIs > LambdaCalc (`*`abc123`*`).`

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

5. Wählen Sie Modell erstellen aus.

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

Um die Anforderungsvalidierung für die POST-Methode zu aktivieren

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

   Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **`$default`** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
6. Wählen Sie für Modell Eingabe aus.
7. Wählen Sie Speichern.

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

So testen Sie die **`POST`**-Methode

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

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

3. Wählen Sie Test aus.

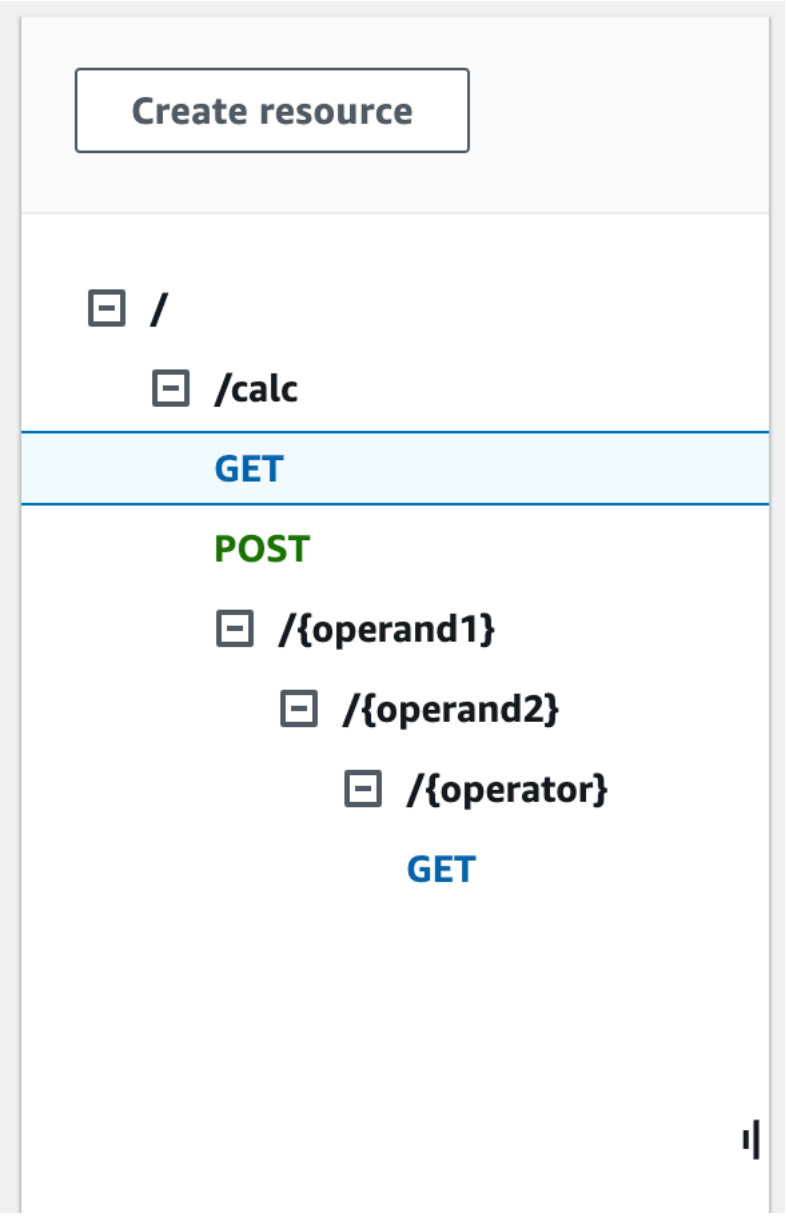
   Die Ausgabe sollte folgendermaßen aussehen:

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

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

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

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



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

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

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

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

Einrichten einer Integrationsmethode

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

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

Erstellen einer Zuweisungsvorlage

1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.

3. Klicken Sie auf Zuordnungsvorlage hinzufügen.
4. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
5. Geben Sie für Vorlagentext den folgenden Code ein:

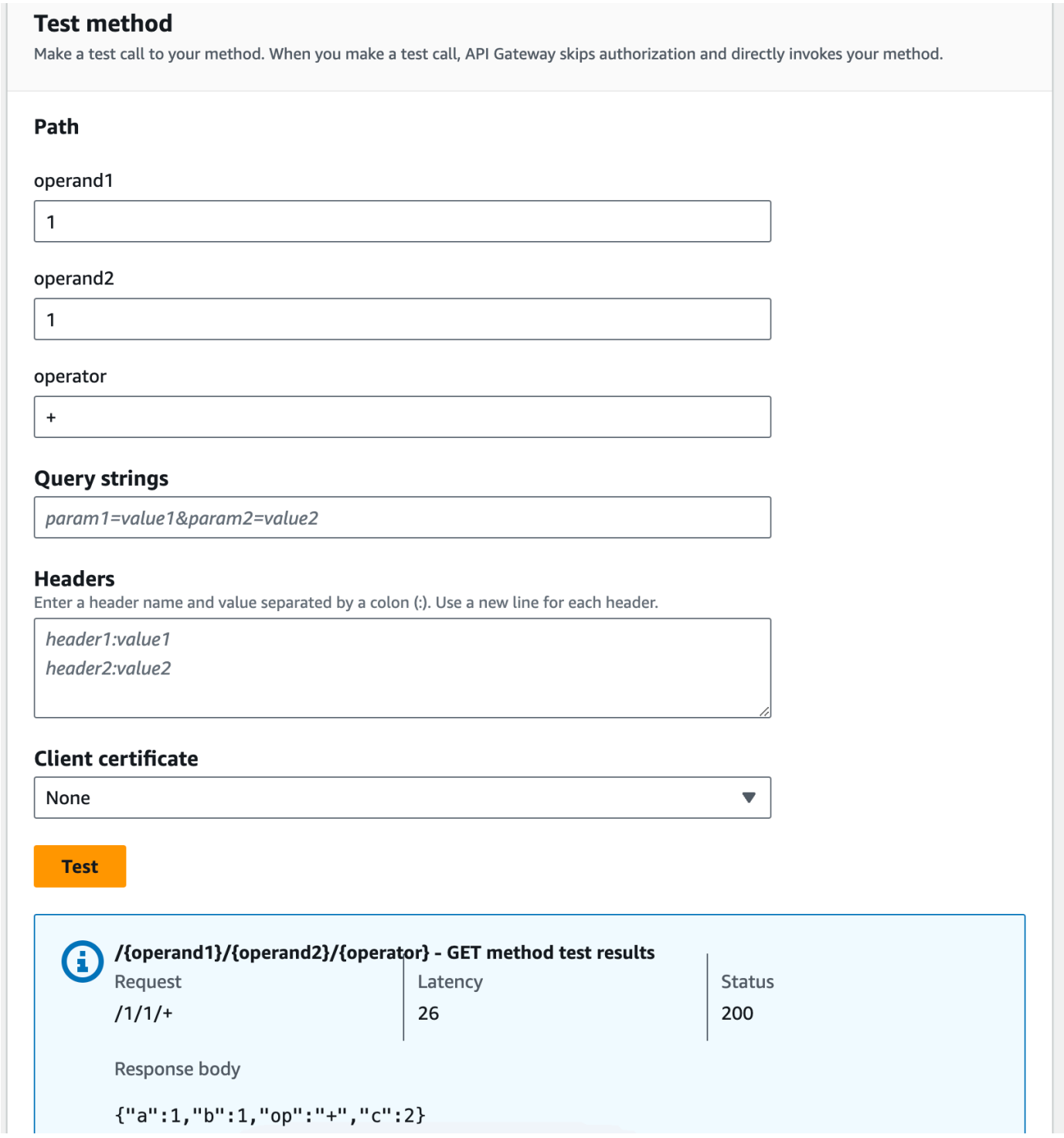
```
{
   "a": "$input.params('operand1')",
   "b": "$input.params('operand2')",
   "op":
 #if($input.params('operator')=='%2F')"/"#{else}"$input.params('operator')"#end
}
```

6. Wählen Sie Speichern.

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

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

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



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

Standardmäßig wird dem Methodenantworttext ein leeres Modell zugeordnet. Dadurch wird der Integrationsantworttext ohne Zuweisung übergeben. Wenn Sie jedoch ein SDK für eine stark

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

Erstellen einer Methodenantwort

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

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

Erstellen einer Zuweisungsvorlage

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

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

```
}
```

5. Wählen Sie Speichern.

Testen der Zuweisungsvorlage

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

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

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

So stellen Sie die API bereit

1. Klicken Sie auf Deploy API.
2. Wählen Sie für Stufe die Option Neue Stufe aus.

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

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

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

OpenAPI 2.0

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

```
        ],
        "parameters": [
          {
            "name": "operand2",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operator",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operand1",
            "in": "query",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            },
            "headers": {
              "operand_1": {
                "type": "string"
              },
              "operand_2": {
                "type": "string"
              },
              "operator": {
                "type": "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator": "Validate query string parameters
and headers",
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
```

```
          "responses": {
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.operator": "integration.response.body.op",
                "method.response.header.operand_2": "integration.response.body.b",
                "method.response.header.operand_1": "integration.response.body.a"
              },
              "responseTemplates": {
                "application/json": "#set($res = $input.path('$'))\n{\n    \"result
\": \"$res.a, $res.b, $res.op => $res.c\",\n  \"a\" : \"$res.a\",\n  \"b\" :
 \"$res.b\",\n  \"op\" : \"$res.op\",\n  \"c\" : \"$res.c\"\n}"
              }
            }
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n    \"a\":  \"$input.params('operand1')\",\n
 \"b\":  \"$input.params('operand2')\", \n    \"op\": \"$input.params('operator')\"
  \n}"
          },
          "type": "aws"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Input"
            }
          }
        ],
```

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

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

```
          "type": "aws"
        }
      }
    }
  },
  "definitions": {
    "Input": {
      "type": "object",
      "required": [
        "a",
        "b",
        "op"
      ],
      "properties": {
        "a": {
          "type": "number"
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string",
          "description": "binary op of ['+', 'add', '-', 'sub', '*', 'mul', '%2F',
'div']"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
```

```
          "$ref": "#/definitions/Output"
        }
      },
      "title": "Result"
    }
  },
  "x-amazon-apigateway-request-validators": {
    "Validate body": {
      "validateRequestParameters": false,
      "validateRequestBody": true
    },
    "Validate query string parameters and headers": {
      "validateRequestParameters": true,
      "validateRequestBody": false
    }
  }
}
```

Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API Gateway

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

- Bereitstellung von GET bei der Stammressource der API, um alle Amazon S3-Buckets eines Aufrufers aufzulisten.
- Bereitstellung von GET bei einer Folder-Ressource, um eine Liste aller Objekte in einem Amazon S3-Bucket anzuzeigen.
- Bereitstellung von GET bei einer Folder/Item-Ressource, um ein Objekt von einem Amazon S3-Bucket anzuzeigen oder herunterzuladen.

Unter Umständen sollten Sie die Beispiel-API wie in OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy gezeigt als Amazon-S3-Proxy importieren. Dieses Beispiel enthält weitere exponierte Methoden. Anweisung zum Importieren einer API mithilfe der OpenAPI-Definition finden Sie unter Konfigurieren einer REST-API mit OpenAPI.

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

Themen

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

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

Einer IAM-Rolle müssen geeignete IAM-Richtlinien zugeordnet sein, damit die API erforderliche Amazon-S3-Aktionen aufrufen kann.

Um die AWS Service-Proxy-Ausführungsrolle zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Roles.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS Dienst aus, wählen Sie dann API Gateway aus und wählen Sie Erlaubt API Gateway, Logs in Logs zu CloudWatch übertragen.
5. Klicken Sie auf Weiter und dann erneut auf Weiter.
6. Geben Sie für Role name (Rollenname) den Namen **`APIGatewayS3ProxyPolicy`** ein und klicken Sie auf Create role (Rolle erstellen).

7. Wählen Sie in der Liste Roles die Rolle aus, die Sie gerade erstellt haben. Möglicherweise müssen Sie scrollen oder die Rolle über die Suchleiste finden.
8. Wählen Sie für die ausgewählte Rolle die Registerkarte Berechtigungen hinzufügen aus.
9. Wählen Sie in der Dropdown-Liste Berechtigungen anfügen aus.
10. Geben Sie im Suchfeld **`AmazonS3FullAccess`** ein und wählen Sie Berechtigungen hinzufügen aus.

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

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

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

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

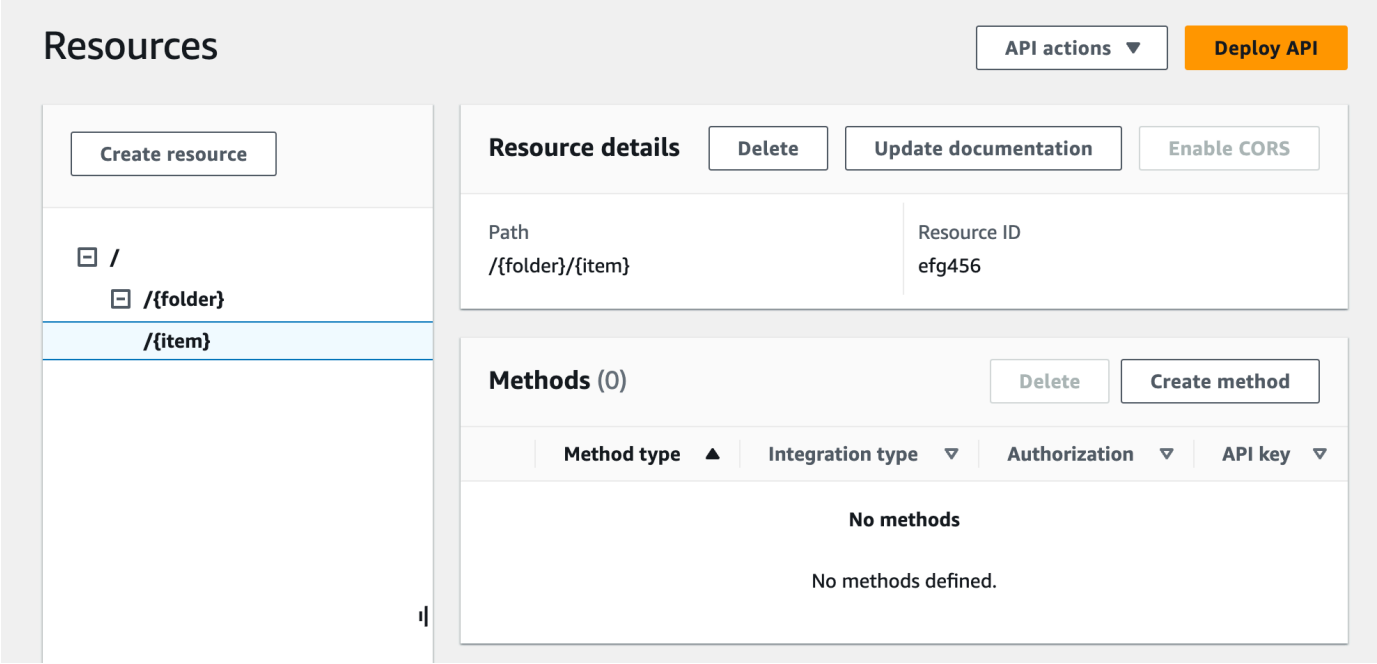
> **Note**
>
> Beim Zugriff auf Objekte, deren Objektschlüssel `/` oder andere Sonderzeichen enthält, müssen diese Zeichen URL-kodiert sein. Beispielsweise sollte `test/test.txt` zu `test%2Ftest.txt` kodiert werden.

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

1. Erstellen Sie in derselben Weise, in der AWS-Region Sie Ihren Amazon S3 S3-Bucket erstellt haben, eine API mit dem Namen myS3. Die Stammressource dieser API (/) stellt den Amazon S3-Service dar. In diesem Schritt erstellen Sie zwei zusätzliche Ressourcen: /{folder} und /{item}.
2. Wählen Sie die Stammressource für die API aus und klicken Sie dann auf Ressource erstellen.
3. Die Proxy-Ressource bleibt ausgeschaltet.

4. Wählen Sie für Ressourcenpfad / aus.
5. Geben Sie für Resource name (Ressourcenname) **`{folder}`** ein.
6. CORS (Cross Origin Resource Sharing) bleibt unmarkiert.
7. Wählen Sie Create Resource (Ressource erstellen) aus.
8. Wählen Sie die /{folder}-Ressource aus und klicken Sie dann auf Ressource erstellen.
9. Wiederholen Sie die Schritte oben, um eine untergeordnete Ressource für /{folder} mit dem Namen **`{item}`** zu erstellen.

   Die endgültige API sollte wie folgt aussehen:



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

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

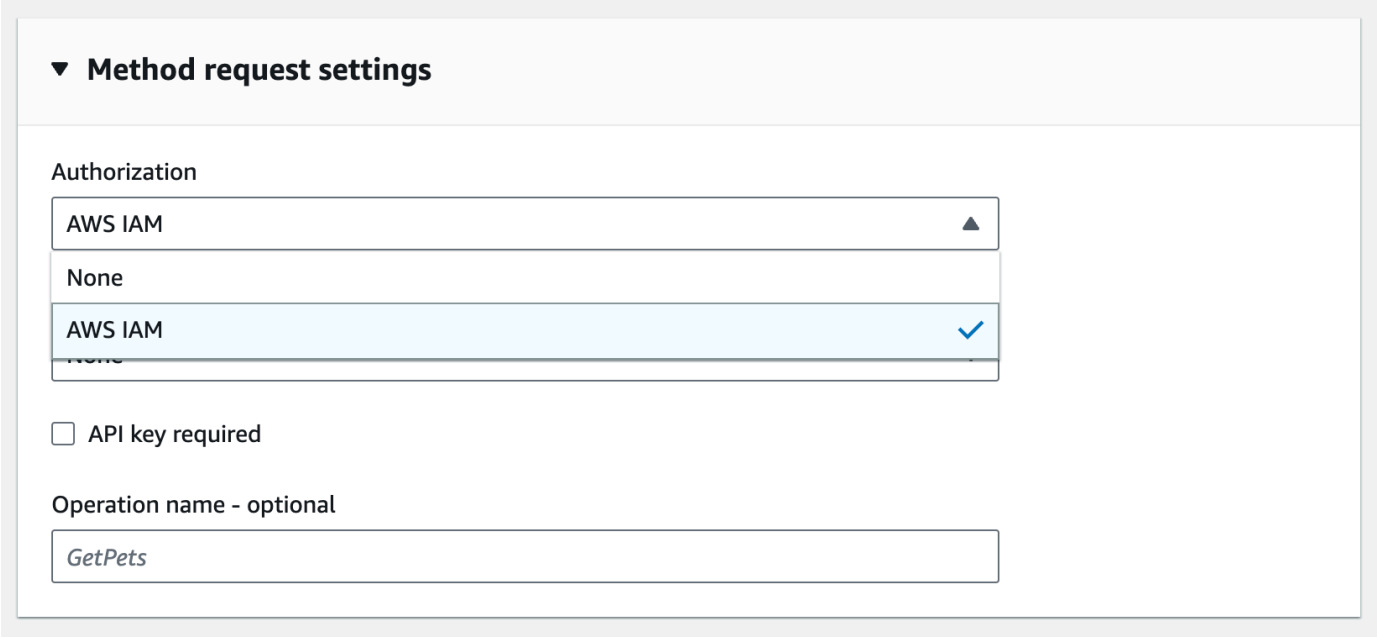
So erstellen und initialisieren Sie die **`GET /`**-Methode der API

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

4. Wählen Sie für den Ort aus AWS-Region, AWS-Region an dem Sie Ihren Amazon S3 S3-Bucket erstellt haben.
5. Wählen Sie für AWS-Service Amazon Simple Storage Service aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Für HTTP-Methode wählen Sie GET aus.
8. Wählen Sie für Aktionstyp Pfadüberschreibung verwenden aus.

   Wenn der Pfad überschrieben wird, leitet API Gateway die Client-Anfrage an Amazon S3 als die entsprechende Amazon S3-REST-API-Path-Style-Anfrage weiter, worin eine Amazon S3-Ressource durch den Ressourcen-Pfad des `s3-host-name/bucket/key`-Musters ausgedrückt wird. API Gateway legt den `s3-host-name` fest und gibt den Client, der in `bucket` und `key` spezifiziert ist, vom Client an Amazon S3 weiter.
9. Geben Sie für Pfadüberschreibung / ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN für **`APIGatewayS3ProxyPolicy`** ein.
11. Wählen Sie Einstellungen für Methodenanfragen aus.

    Sie verwenden die Einstellungen für Methodenanfragen, um zu steuern, wer diese Methode Ihrer API aufrufen kann.
12. Wählen Sie im Dropdown-Menü `AWS_IAM` für Autorisierung aus.



13. Wählen Sie Methode erstellen aus.

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

Damit Ihre API erfolgreiche Antworten und Ausnahmen ordnungsgemäß an den Aufrufer zurückgibt, deklarieren Sie die Antworten 200, 400 und 500 in Method Response. Sie verwenden die Standardzuordnung für 200 Antworten, sodass Backend-Antworten mit dem hier nicht deklarierten Statuscode als 200 Antworten an den Aufrufer zurückgegeben werden.

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

1. Klicken Sie auf der Registerkarte Methodenantwort unter Antwort 200 auf Bearbeiten.
2. Klicken Sie auf Header hinzufügen und gehen Sie wie folgt vor:

   a. Für Header-Name geben Sie **`Content-Type`** ein.

   b. Wählen Sie Add header.

   Wiederholen Sie diese Schritte, um einen **`Timestamp`**-Header und einen **`Content-Length`**-Header zu erstellen.
3. Wählen Sie Speichern.
4. Klicken Sie auf der Registerkarte Methodenantwort unter Methodenantworten auf Antwort erstellen.
5. Geben Sie als HTTP-Statuscode 400 ein.

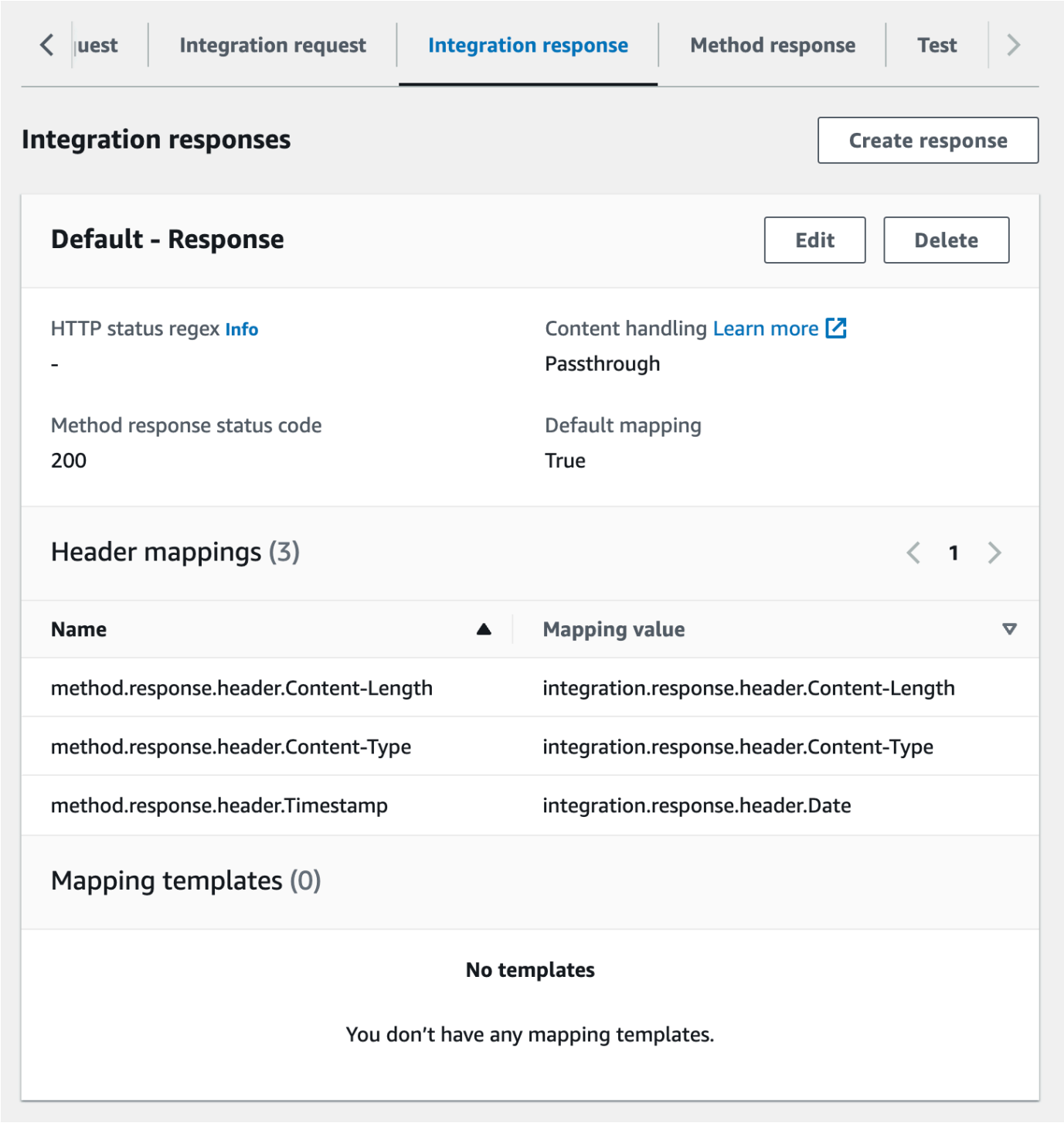
   Für diese Antwort legen Sie keine Header fest.
6. Wählen Sie Speichern.
7. Wiederholen Sie die folgenden Schritte, um die Antwort 500 zu erstellen.

   Für diese Antwort legen Sie keine Header fest.

Da die erfolgreiche Integrationsantwort von Amazon S3 die Bucket-Liste als XML-Nutzlast zurückgibt und die Standardmethodenantwort von API Gateway eine JSON-Nutzlast zurückgibt, müssen Sie den Wert des Backend-Header-Parameters Content-Type dem Frontend-Gegenstück zuordnen. Andernfalls erhält der Client `application/json` als Content-Type, wenn der Antworttext eigentlich eine XML-Zeichenfolge ist. Das folgende Verfahren zeigt, wie diese Einrichtung erfolgt. Darüber hinaus möchten Sie dem Client auch andere Header-Parameter wie Datum und Inhaltslänge anzeigen.

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

1. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
2. Geben Sie für den Header Content-Length den Wert **`integration.response.header.Content-Length`** als Zuordnungswert ein.
3. Geben Sie für den Header Content-Type den Wert **`integration.response.header.Content-Type`** als Zuordnungswert ein.
4. Geben Sie für den Header Timestamp den Wert **`integration.response.header.Date`** als Zuordnungswert ein.
5. Wählen Sie Speichern. Das Ergebnis sollte in etwa wie folgt aussehen:



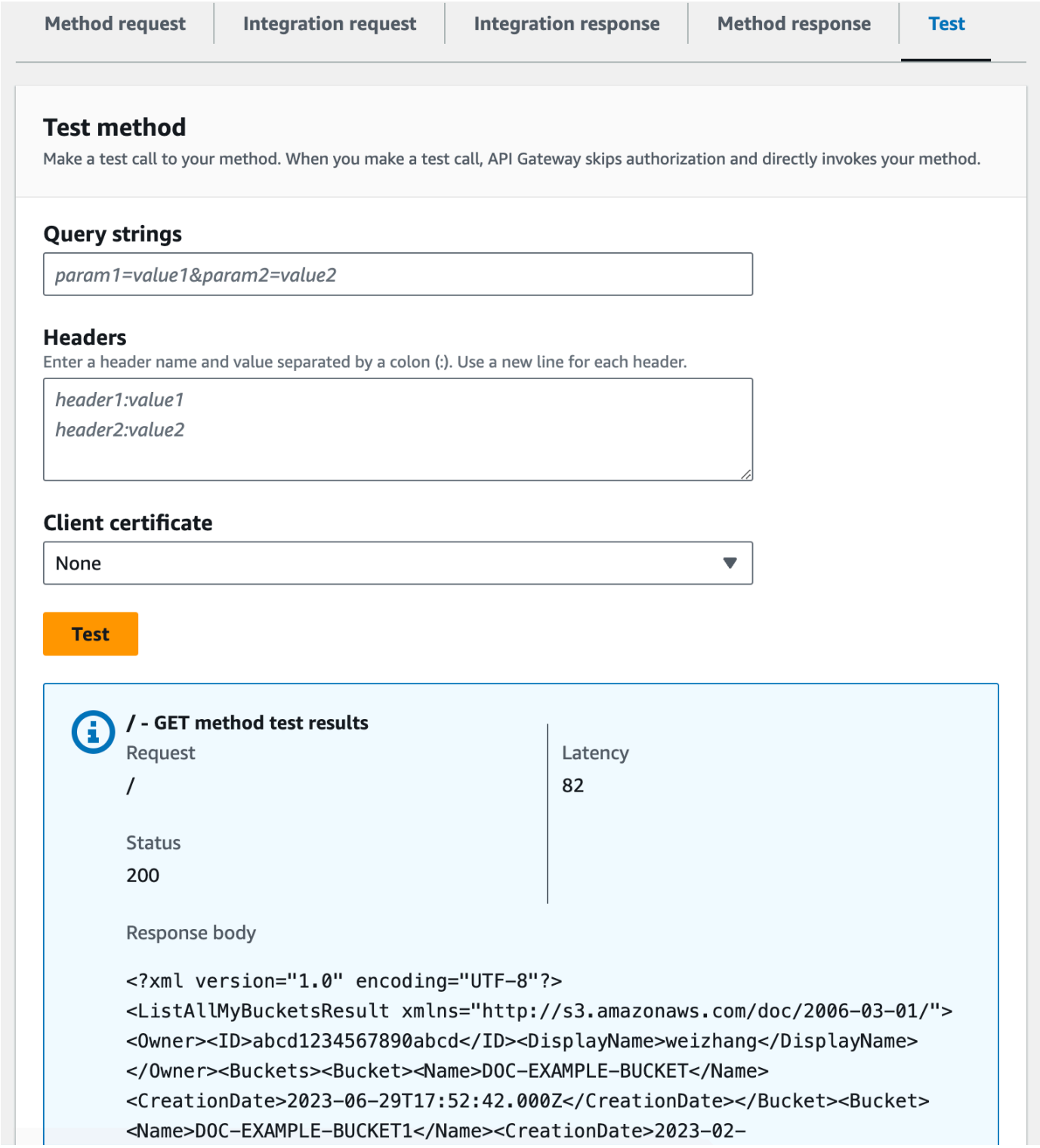
6. Klicken Sie auf der Registerkarte Integrationsantwort unter Integrationsantworten auf Antwort erstellen.
7. Machen Sie für HTTP status regex (HTTP-Status-RegEx) den Eintrag **`4\d{2}`**. Dadurch werden alle 4xx-HTTP-Antwortstatuscodes der Methodenantwort zugeordnet.
8. Wählen Sie für Statuscode der Methodenantwort **`400`** aus.

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

Es hat sich bewährt, die API zu testen, die Sie bisher konfiguriert haben.

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

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


Method request
Integration request
Integration response
Method response
Test
Test method
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.
Query strings
param1=value1&param2=value2
Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2
Client certificate
None
Test
/ - GET method test results
Request
/
Status
200
Latency
82
Response body
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Owner><ID>abcd1234567890abcd</ID><DisplayName>weizhang</DisplayName>
</Owner><Buckets><Bucket><Name>DOC-EXAMPLE-BUCKET</Name>
<CreationDate>2023-06-29T17:52:42.000Z</CreationDate></Bucket><Bucket>
<Name>DOC-EXAMPLE-BUCKET1</Name><CreationDate>2023-02-

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

Um mit einem Amazon S3 S3-Bucket zu arbeiten, machen Sie die `GET` Methode für die Ressource/{folder} verfügbar, um Objekte in einem Bucket aufzulisten. Die Anweisungen sind ähnlich wie diejenigen, die unter beschrieben sin Bereitstellen einer API-Methode zur Auflistung der Amazon S3-Buckets des Aufrufers. Sie können die Beispiel-API hier für weitere Methoden importieren, OpenAPI-Definitionen der Beispiel-API als Amazon-S3-Proxy.

Exponieren der GET-Methoden in einer Ordnerressource

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

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

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

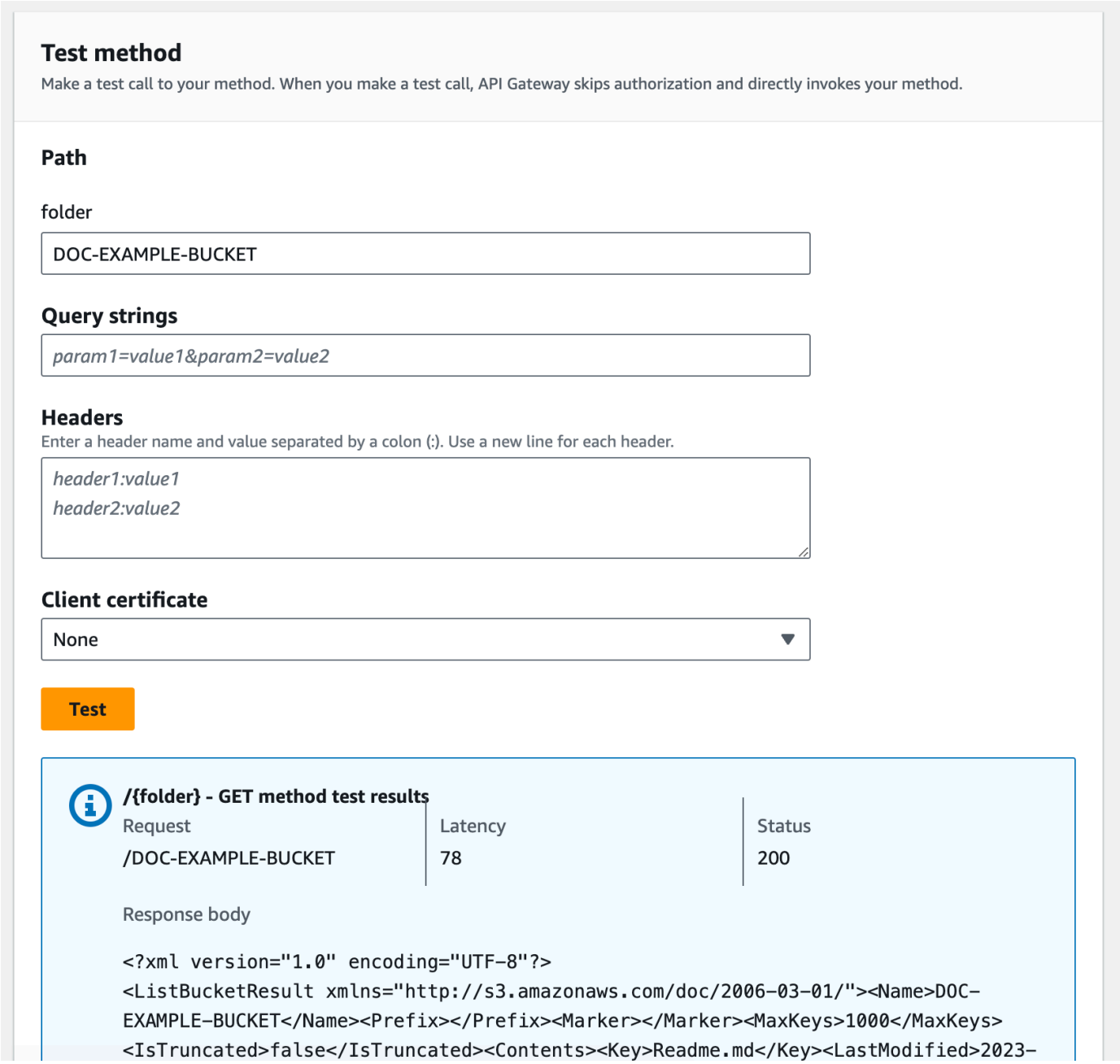
1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Wählen Sie URL-Pfadparameter aus und klicken Sie dann auf Pfadparameter hinzufügen.
3. Geben Sie unter Name **`bucket`** ein.
4. Geben Sie für Zugeordnet von **`method.request.path.folder`**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

Als Nächstes testen Sie Ihre API.

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

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

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



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

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

Exponieren der GET-Methode in einer item-Ressource

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

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

In diesem Schritt führen Sie folgende Aufgaben aus:

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

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

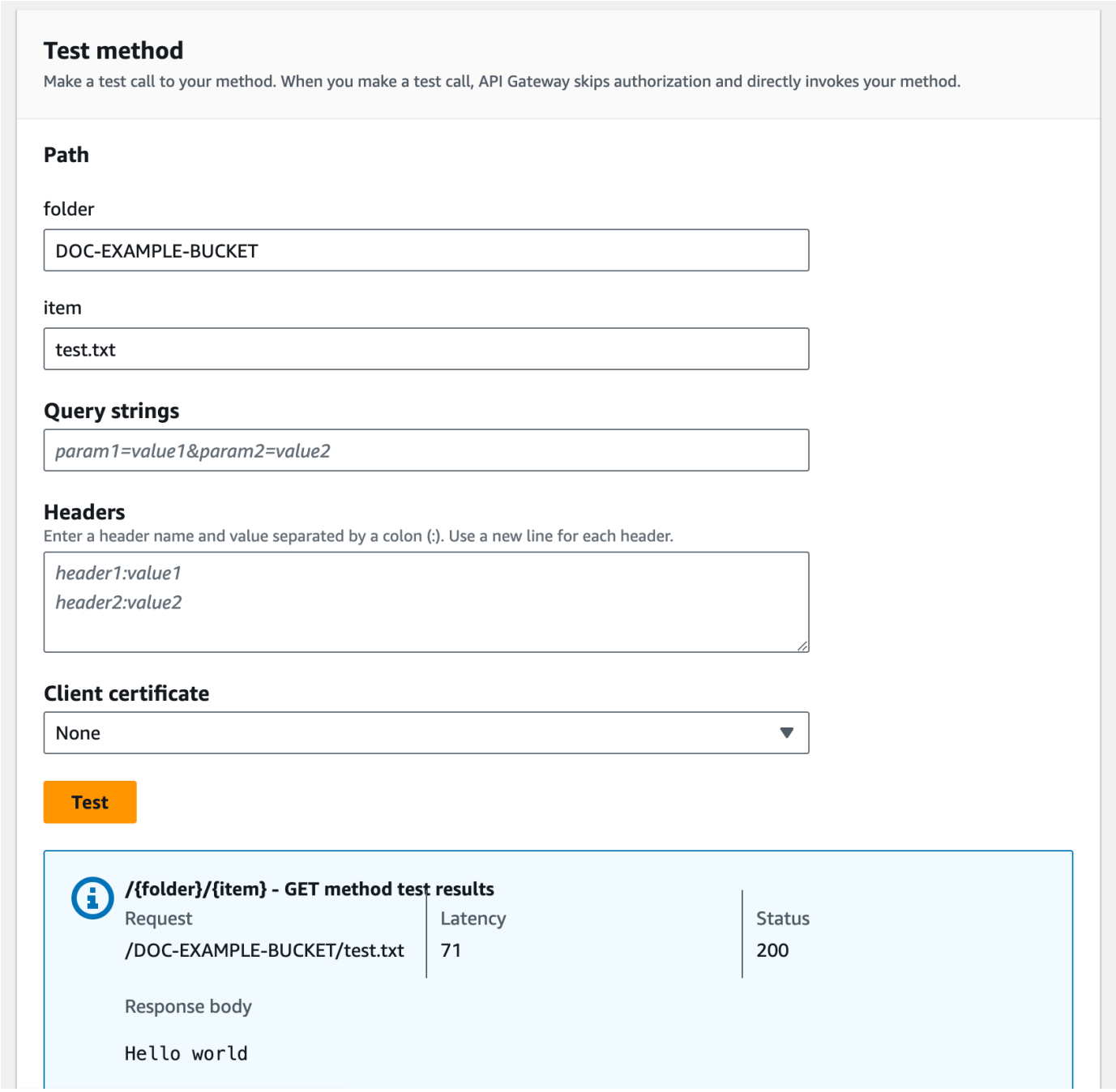
1. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
2. Klicken Sie auf URL-Pfadparameter.
3. Klicken Sie auf Pfadparameter hinzufügen.
4. Geben Sie unter Name **`bucket`** ein.
5. Geben Sie für Zugeordnet von **`method.request.path.folder`**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

6. Klicken Sie auf Pfadparameter hinzufügen.
7. Geben Sie unter Name **`object`** ein.
8. Geben Sie für Zugeordnet von **`method.request.path.item`** ein.
9. Wählen Sie Speichern.

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

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

   Der Antworttext enthält den Inhalt des Elements.



Die Anforderung gibt den Klartext („Hello world“) als Inhalt der angegebenen Datei (test.txt) in dem entsprechenden Amazon-S3-Bucket (DOC-EXAMPLE-BUCKET) zurück.

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

Herunterladen oder Hochladen von Binärdateien von S3

1. Registrieren Sie die Medientypen der betroffenen Datei bei den APIs binaryMediaTypes. Sie können Sie von der Konsole aus erledigen:

   a. Wählen Sie API-Einstellungen für die API aus.
   b. Klicken Sie unter Binäre Medientypen auf Medientypen verwalten.
   c. Klicken Sie auf Binären Medientyp hinzufügen und geben Sie dann den erforderlichen Medientyp ein, z. B. `image/png`.
   d. Wählen Sie zum Speichern der Einstellung Save Changes (Änderungen speichern) aus.
2. Fügen Sie den Header `Content-Type` (zum Hochladen) und/oder `Accept` (zum Herunterladen) zur Methodenanfrage hinzu, um den Client aufzufordern, den erforderlichen binären Medientyp anzugeben, und weisen Sie sie der Integrationsanfrage zu.
3. Setzen Sie Content Handling in der Integrationsanfrage auf `Passthrough` (für das Hochladen), und in einer Integrationsantwort (für das Herunterladen). Stellen Sie sicher, dass keine Mapping-Vorlage für den betroffenen Inhaltstyp definiert ist. Weitere Informationen finden Sie unter Integrations-Pass-Through-Verhalten und Auswählen einer VTL-Mapping-Vorlage.

Die maximale Größe der Nutzlast ist 10 MB. Siehe API Gateway-Kontingente für die Konfiguration und Ausführung einer REST-API.

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

Weitere Informationen zur Unterstützung von Binärdateien in API Gateway finden Sie in Inhaltstypkonvertierungen in API Gateway.

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

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

- Bereitstellung von GET bei der Stammressource der API, um alle Amazon S3-Buckets eines Aufrufers aufzulisten.
- Bereitstellung von GET bei einer Folder-Ressource, um eine Liste aller Objekte in einem Amazon S3-Bucket anzuzeigen.

- Bereitstellung von PUT bei einer Folder-Ressource, um einen Bucket zu Amazon S3 hinzuzufügen.
- Bereitstellung von DELETE bei einer Folder-Ressource, um einen Bucket aus Amazon S3 zu entfernen.
- Bereitstellung von GET bei einer Folder/Item-Ressource, um ein Objekt von einem Amazon S3-Bucket anzuzeigen oder herunterzuladen.
- Bereitstellung von PUT bei einer Folder/Item-Ressource, um ein Objekt zu einem Amazon S3-Bucket hochzuladen.
- Bereitstellung von HEAD bei einer Folder/Item-Ressource, um Objekt-Metadaten in einem Amazon S3-Bucket abzurufen.
- Bereitstellung von DELETE bei einer Folder/Item-Ressource, um ein Objekt aus einem Amazon S3-Bucket zu entfernen.

Anweisung zum Importieren einer API mithilfe der OpenAPI-Definition finden Sie unter Konfigurieren einer REST-API mit OpenAPI.

Eine Anleitung zur Erstellung einer ähnlichen API finden Sie unter Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API Gateway.

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

OpenAPI 2.0

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

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

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

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

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

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

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

```
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "DELETE",
          "type": "aws"
        }
      }
    },
    "/{folder}/{item}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "content-type": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
```

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

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

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

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

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

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

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

OpenAPI 3.0

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

```
        }
      } ],
      "responses" : {
        "400" : {
          "description" : "400 response",
          "content" : { }
        },
        "500" : {
          "description" : "500 response",
          "content" : { }
        },
        "200" : {
          "description" : "200 response",
          "headers" : {
            "Content-Length" : {
              "schema" : {
                "type" : "string"
              }
            },
            "Date" : {
              "schema" : {
                "type" : "string"
              }
            },
            "Content-Type" : {
              "schema" : {
                "type" : "string"
              }
            }
          },
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Empty"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration" : {
        "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "httpMethod" : "GET",
        "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
        "responses" : {
```

```
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Date" : "integration.response.header.Date",
                "method.response.header.Content-Length" :
"integration.response.header.content-length"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "put" : {
        "parameters" : [ {
          "name" : "Content-Type",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
```

```
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "PUT",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
              }
            },
```

```
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder",
            "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "delete" : {
        "parameters" : [ {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Date" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
```

```
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "DELETE",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Date" : "integration.response.header.Date"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      }
    },
    "/{folder}/{item}" : {
      "get" : {
        "parameters" : [ {
          "name" : "item",
          "in" : "path",
          "required" : true,
```

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

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

```
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "PUT",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
```

```
            "statusCode" : "400"
          },
          "default" : {
            "statusCode" : "200",
            "responseParameters" : {
              "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
              "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
            }
          },
          "5\\d{2}" : {
            "statusCode" : "500"
          }
        },
        "requestParameters" : {
          "integration.request.path.object" : "method.request.path.item",
          "integration.request.path.bucket" : "method.request.path.folder",
          "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "aws"
      }
    },
    "delete" : {
      "parameters" : [ {
        "name" : "item",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      }, {
        "name" : "folder",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "responses" : {
        "400" : {
          "description" : "400 response",
```

```
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "DELETE",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200"
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
```

```
        "requestParameters" : {
          "integration.request.path.object" : "method.request.path.item",
          "integration.request.path.bucket" : "method.request.path.folder"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "aws"
      }
    },
    "head" : {
      "parameters" : [ {
        "name" : "item",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      }, {
        "name" : "folder",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "responses" : {
        "400" : {
          "description" : "400 response",
          "content" : { }
        },
        "500" : {
          "description" : "500 response",
          "content" : { }
        },
        "200" : {
          "description" : "200 response",
          "headers" : {
            "Content-Length" : {
              "schema" : {
                "type" : "string"
              }
            },
            "Content-Type" : {
              "schema" : {
                "type" : "string"
```

```
              }
            }
          },
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Empty"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration" : {
        "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "httpMethod" : "HEAD",
        "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
        "responses" : {
          "4\\d{2}" : {
            "statusCode" : "400"
          },
          "default" : {
            "statusCode" : "200",
            "responseParameters" : {
              "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
              "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
            }
          },
          "5\\d{2}" : {
            "statusCode" : "500"
          }
        },
        "requestParameters" : {
          "integration.request.path.object" : "method.request.path.item",
          "integration.request.path.bucket" : "method.request.path.folder"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "aws"
      }
    }
  },
  "/" : {
    "get" : {
```

```
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Timestamp" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "GET",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path//",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
```

```
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
 "integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
 "integration.response.header.Content-Length",
                "method.response.header.Timestamp" :
 "integration.response.header.Date"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Empty" : {
        "title" : "Empty Schema",
        "type" : "object"
      }
    }
  }
}
```

Aufrufen der API mit einem REST-API-Client

Als end-to-end Tutorial zeigen wir nun, wie die API mithilfe von Postman aufgerufen wird, das die IAM-Autorisierung unterstützt. AWS

So rufen Sie die Amazon S3-Proxy-API mit Postman auf

1. Stellen Sie die API bereit (oder erneut bereit). Notieren Sie die Basis-URL der API, die neben Invoke URL oben im Stage Editor angezeigt wird.
2. Starten Sie Postman.

3. Wählen Sie Autorisierung und anschließend `AWS Signature`. Geben Sie die Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel Ihres IAM-Benutzers in die jeweiligen AccessKeySecretKeyEingabefelder ein. Geben Sie die AWS Region, in der Ihre API bereitgestellt wird, in das Textfeld AWS Region ein. Geben Sie `execute-api` in das Eingabefeld Service Name ein.

   Sie können ein Schlüsselpaar über die Registerkarte Security Credentials (Sicherheitsanmeldeinformationen) von Ihrem IAM-Benutzerkonto in der IAM Management Console erstellen.

4. So fügen Sie einen Bucket mit dem Namen `apig-demo-5` zu Ihrem Amazon S3-Konto in der Region *`{region}`* hinzu:

   > **Note**
   >
   > Stellen Sie sicher, dass der Bucket-Name global einzigartig ist.

   a. Wählen Sie in der Dropdown-Methodenliste PUT aus und geben Sie die Methoden-URL ein (`https://`*`api-id`*`.execute-api.`*`aws-region`*`.amazonaws.com/`*`stage`*`/`*`folder-name`*)

   b. Geben Sie bei `Content-Type` den Header-Wert `application/xml` ein. Sie müssen möglicherweise vorhandene Header löschen, bevor Sie den Content-Type angeben können.

   c. Wählen Sie das Menüelement Body aus, und geben Sie das folgende XML-Fragment als Anforderungstext ein:

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

   d. Wählen Sie Senden aus, um die Anforderung zu senden. Bei Erfolg erhalten Sie die Antwort `200 OK` mit einer leeren Nutzlast.

5. Um zu einem Bucket eine Textdatei hinzuzufügen, befolgen Sie die obigen Anweisungen. Wenn Sie den Bucket-Namen **`apig-demo-5`** für `{folder}` und den Dateinamen für **`Readme.txt`** für `{item}` in der URL angeben und die Textzeichenfolge **`Hello, World!`** als Dateiinhalt (und damit als Anforderungsnutzlast) angeben, wird die Anforderung

   ```
   PUT /S3/apig-demo-5/Readme.txt HTTP/1.1
   ```

```
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T062647Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-
api/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=ccadb877bdb0d395ca38cc47e18a0d76bb5eaf17007d11e40bf6fb63d28c705b
Cache-Control: no-cache
Postman-Token: 6135d315-9cc4-8af8-1757-90871d00847e

Hello, World!
```

Wenn alles einwandfrei verläuft, erhalten Sie die Antwort `200 OK` mit einer leeren Nutzlast.

6. Um den Inhalt der Datei `Readme.txt` abzurufen, die wir soeben zum `apig-demo-5`-Bucket hinzugefügt haben, erstellen wir eine GET-Anforderung, die wie folgt aussieht:

```
GET /S3/apig-demo-5/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T063759Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=ba09b72b585acf0e578e6ad02555c00e24b420b59025bc7bb8d3f7aed1471339
Cache-Control: no-cache
Postman-Token: d60fcb59-d335-52f7-0025-5bd96928098a
```

Bei Erfolg erhalten Sie die Antwort `200 OK` mit der Textzeichenfolge `Hello, World!` als Nutzlast.

7. Um Elemente im `apig-demo-5`-Bucket aufzulisten, senden Sie die folgende Anforderung:

```
GET /S3/apig-demo-5 HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T064324Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=4ac9bd4574a14e01568134fd16814534d9951649d3a22b3b0db9f1f5cd4dd0ac
Cache-Control: no-cache
Postman-Token: 9c43020a-966f-61e1-81af-4c49ad8d1392
```

Bei Erfolg erhalten Sie die Antwort `200 OK` mit einer XML-Nutzlast mit einem einzelnen Element im angegebenen Bucket, es sei denn, Sie haben noch mehr Dateien zu dem Bucket hinzugefügt, bevor Sie diese Anforderung gesendet haben.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>apig-demo-5</Name>
    <Prefix></Prefix>
    <Marker></Marker>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>Readme.txt</Key>
        <LastModified>2016-10-15T06:26:48.000Z</LastModified>
        <ETag>"65a8e27d8879283831b664bd8b7f0ad4"</ETag>
        <Size>13</Size>
        <Owner>
            <ID>06e4b09e9d...603addd12ee</ID>
            <DisplayName>user-name</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

> **Note**
>
> Um ein Bild hoch- oder herunterzuladen, müssen Sie für die Inhaltsbehandlung die Option CONVERT_TO_BINARY auswählen.

Tutorial: REST-API als Amazon Kinesis-Proxy in API Gateway erstellen

Auf dieser Seite wird beschrieben, wie eine REST-API durch Integration des `AWS`-Typs für den Zugriff auf Kinesis erstellt und konfiguriert wird.

> **Note**
>
> Um Ihre API Gateway-API in Kinesis zu integrieren, müssen Sie eine Region wählen, in der sowohl das API Gateway als auch die Kinesis-Services verfügbar sind. Informationen zur Verfügbarkeit der Regionen finden Sie unter Service-Endpunkte und Kontingente.

Zur Veranschaulichung erstellen wir eine Beispiel-API, so dass ein Client die folgenden Aufgaben ausführen kann:

1. Verfügbare Streams des Benutzers in Kinesis auflisten
2. Erstellen, Beschreiben oder Löschen eines bestimmten Streams
3. Lesen von Datensätzen aus einem oder Schreiben von Datensätzen in einen bestimmten Stream

Für die Ausführung der obigen Aufgaben stellt die API Methoden für verschiedene Ressourcen bereit, um Folgendes aufzurufen:

1. Die `ListStreams`-Aktion in Kinesis
2. Die Aktion `CreateStream`, `DescribeStream` oder `DeleteStream`
3. Die `GetRecords`- oder `PutRecords`-Aktion (einschließlich `PutRecord`) in Kinesis

Genauer gesagt wird die API wie folgt erstellt:

- Machen Sie eine HTTP-GET-Methode für die `/streams` API-Ressource verfügbar und integrieren Sie die Methode in die ListStreamsAktion in Kinesis, um die Streams im Konto des Aufrufers aufzulisten.
- Machen Sie eine HTTP-POST-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die CreateStreamAktion in Kinesis, um einen benannten Stream im Konto des Aufrufers zu erstellen.
- Machen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die DescribeStreamAktion in Kinesis, um einen benannten Stream im Konto des Aufrufers zu beschreiben.
- Machen Sie eine HTTP-DELETE-Methode für die `/streams/{stream-name}` API-Ressource verfügbar und integrieren Sie die Methode in die DeleteStreamAktion in Kinesis, um einen Stream im Konto des Anrufers zu löschen.

- Machen Sie eine HTTP-PUT-Methode für die `/streams/{stream-name}/record` API-Ressource verfügbar und integrieren Sie die Methode in die PutRecordAktion in Kinesis. Dadurch kann der Client einen einzelnen Datensatz zum benannten Stream hinzufügen.
- Machen Sie eine HTTP-PUT-Methode für die `/streams/{stream-name}/records` API-Ressource verfügbar und integrieren Sie die Methode in die PutRecordsAktion in Kinesis. Dadurch kann der Client eine Liste mit Datensätzen zum benannten Stream hinzufügen.
- Stellen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}/records` API-Ressource bereit und integrieren Sie die Methode in die GetRecordsAktion in Kinesis. Auf diese Weise kann der Client Datensätze im benannten Stream mit einem angegebenen gemeinsamen Iterator auflisten. Ein gemeinsamer Iterator gibt die gemeinsame Position an, ab der Datensätze sequenziell ausgelesen werden sollen.
- Stellen Sie eine HTTP-GET-Methode für die `/streams/{stream-name}/sharditerator` API-Ressource bereit und integrieren Sie die Methode in die GetShardIteratorAktion in Kinesis. Diese Hilfsmethode muss der `ListStreams`-Aktion in Kinesis zur Verfügung gestellt werden.

Sie können die hier vorgestellten Anweisungen auf andere Kinesis-Aktionen anwenden. Die vollständige Liste der Kinesis-Aktionen finden Sie unter Amazon Kinesis API-Referenz.

Anstatt die API Gateway-Konsole zur Erstellung der Beispiel-API zu verwenden, können Sie die Beispiel-API mit der API Gateway Import API in API Gateway importieren. Informationen zur Verwendung der Import-API finden Sie unter Konfigurieren einer REST-API mit OpenAPI.

Erstellen Sie eine IAM-Rolle und Richtlinien für den Zugriff der API auf Kinesis

Einer IAM-Rolle müssen entsprechende IAM-Richtlinien zugeordnet sein, damit die API Kinesis-Aktionen aufrufen kann.

Um die AWS Service-Proxy-Ausführungsrolle zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Roles.
3. Wählen Sie Rolle erstellen aus.
4. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS Dienst aus, wählen Sie dann API Gateway aus und wählen Sie Erlaubt API Gateway, Logs in Logs zu CloudWatch übertragen.
5. Klicken Sie auf Weiter und dann erneut auf Weiter.

6. Geben Sie für Role name (Rollenname) den Namen **`APIGatewayKinesisProxyPolicy`** ein und klicken Sie auf Create role (Rolle erstellen).
7. Wählen Sie in der Liste Roles die Rolle aus, die Sie gerade erstellt haben. Möglicherweise müssen Sie scrollen oder die Rolle über die Suchleiste finden.
8. Wählen Sie für die ausgewählte Rolle die Registerkarte Berechtigungen hinzufügen aus.
9. Wählen Sie in der Dropdown-Liste Berechtigungen anfügen aus.
10. Geben Sie im Suchfeld **`AmazonKinesisFullAccess`** ein und wählen Sie Berechtigungen hinzufügen aus.

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

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

Eine API als Kinesis-Proxy erstellen

Führen Sie die folgenden Schritte aus, um die API in der API Gateway-Konsole zu erstellen.

Um eine API als AWS Service-Proxy für Kinesis zu erstellen

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

   Wenn Sie API Gateway nicht zum ersten Mal verwenden, wählen Sie Create API (API erstellen). Wählen Sie unter REST-API die Option Build (Erstellen) aus.
3. Klicken Sie auf New API (Neue API).
4. Geben Sie unter API name (API-Name) **`KinesisProxy`** ein. Belassen Sie alle Standardwerte für die anderen Felder unverändert.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.

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

Nachdem die API erstellt wurde, zeigt die API Gateway-Konsole die Seite Resources (Ressourcen) an, die nur die Root-Ressource (`/`) der API enthält.

Streams in Kinesis auflisten

Kinesis unterstützt die Aktion `ListStreams` mit dem folgenden REST-API-Aufruf:

```
POST /?Action=ListStreams HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>

{
   ...
}
```

In der oben gezeigten REST-API-Anforderung wird die Aktion im `Action`-Abfrageparameter angegeben. Alternativ können Sie stattdessen die Aktion in einer `X-Amz-Target`-Kopfzeile angeben:

```
POST / HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>
X-Amz-Target: Kinesis_20131202.ListStreams
{
   ...
}
```

In diesem Tutorial verwenden wir den Abfrageparameter, um Aktionen festzulegen.

Um eine Kinesis-Aktion in der API bereitzustellen, fügen Sie eine `/streams`-Ressource zum Stamm der API hinzu. Richten Sie dann eine `GET`-Methode für die Ressource ein und integrieren Sie die Methode mit der `ListStreams`-Aktion von Kinesis.

Das folgende Verfahren beschreibt die Auflistung von Kinesis-Streams mit Hilfe der API Gateway-Konsole.

So listen Sie Kinesis-Streams über die API Gateway-Konsole auf:

1. Wählen Sie die `/`-Ressource aus und klicken Sie dann auf Ressource erstellen.
2. Geben Sie für Resource name (Ressourcenname) **`streams`** ein.
3. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
4. Wählen Sie Create Resource (Ressource erstellen) aus.
5. Wählen Sie die `/streams`-Ressource aus, klicken Sie auf Methode erstellen und gehen Sie dann wie folgt vor:

   a. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.

   > **Note**
   >
   > Das HTTP-Verb für eine Methode, die von einem Client aufgerufen wird, kann sich von dem HTTP-Verb unterscheiden, das für eine Integration für das Backend erforderlich ist. Hier wählen wir `GET` aus, da das Auflisten von Streams intuitiv ein READ-Vorgang ist.

   b. Als Integrationstyp wählen Sie AWS -Dienst aus.

   c. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.

   d. Für AWS-Service wählen Sie Kinesis aus.

   e. Lassen Sie die AWS -Subdomain leer.

   f. Wählen Sie in HTTP method POST.

   > **Note**
   >
   > Wir haben hier `POST` gewählt, weil Kinesis erfordert, dass die Aktion `ListStreams` mit aufgerufen wird.

   g. Für Aktionstyp wählen Sie Aktionsname verwenden aus.

   h. Für Aktionsname geben Sie **`ListStreams`** ein.

   i. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.

j. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.

k. Wählen Sie Methode erstellen aus.

6. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
7. Wählen Sie für Anforderungstext-Pass-Through die Einstellung Wenn keine Vorlagen definiert sind (empfohlen) aus.
8. Klicken Sie auf URL-Abfragezeichenfolgen-Parameter und gehen Sie dann wie folgt vor:

   a. Klicken Sie auf Parameter Anforderungsheader hinzufügen.

   b. Geben Sie unter Name **`Content-Type`** ein.

   c. Geben Sie für Zugeordnet von **`'application/x-amz-json-1.1'`** ein.

   Wir verwenden eine Zuweisung für den Anfrageparameter, um den `Content-Type`-Header auf den statischen Wert `'application/x-amz-json-1.1'` einzustellen und damit Kinesis mitzuteilen, dass die Eingabe in einer bestimmten JSON-Version erfolgt.
9. Wählen Sie Mapping-Vorlagen und dann Mapping-Vorlage hinzufügen aus und gehen Sie dann wie folgt vor:

   a. Geben Sie für Inhaltstyp **`application/json`** ein.

   b. Geben Sie für Vorlage **`{}`** ein.

   c. Wählen Sie Speichern.

   Die ListStreamsAnfrage benötigt eine Nutzlast im folgenden JSON-Format:

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

   Die Eigenschaften sind jedoch optional. Wir haben uns für eine leere JSON-Nutzlast entschieden, um die Standardwerte zu verwenden.
10. Testen Sie die GET-Methode für die /streams-Ressource zum Aufrufen der `ListStreams`-Aktion in Kinesis:

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

Klicken Sie auf Test, um Ihre Methode zu testen.

Wenn Sie bereits zwei Streams mit den Namen „myStream“ und „yourStream“ in Kinesis erstellt haben, gibt der Test im Erfolgsfall die Antwort „200 OK“ mit der folgenden Nutzlast zurück:

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

Stream in Kinesis erstellen, beschreiben und löschen

Das Erstellen, Beschreiben und Löschen eines Streams in Kinesis umfasst jeweils die folgenden Kinesis REST-API-Anfragen:

```
POST /?Action=CreateStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardCount": number,
    "StreamName": "string"
}
```

```
POST /?Action=DescribeStream HTTP/1.1
Host: kinesis.region.domain
...
```

```
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName": "string"
}
```

```
POST /?Action=DeleteStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName":"string"
}
```

Sie können die API so erstellen, dass sie die erforderliche Eingabe als JSON-Nutzlast der Methodenanforderung akzeptiert und die Nutzlast bis zur Integrationsanforderung weiterleitet. Die API wird jedoch geringfügig abgewandelt, um weitere Beispiele für die Datenzuweisung zwischen Methoden- und Integrationsanforderungen sowie Methoden- und Integrationsantworten bereitzustellen.

Wir machen die Methoden`GET`,`POST`, und `Delete` HTTP für eine to-be-named `Stream` Ressource verfügbar. Sie verwenden die Pfadvariable `{stream-name}` als Platzhalter für die Stream-Ressource und integrieren diese API-Methoden in die entsprechenden `DescribeStream`-, `CreateStream`- und `DeleteStream`-Aktionen von Kinesis. Wir benötigen, dass der Client andere Eingabedaten als Header, Abfrageparameter oder Nutzdaten einer Methodenanforderung weitergibt. Wir stellen Mapping-Vorlagen zur Verfügung, um die Daten in die erforderliche Nutzlast der Integrationsanforderung zu transformieren.

Erstellen einer {Stream-name}-Ressource

1. Wählen Sie die Ressource /streams und dann Create resource aus.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Wählen Sie für Ressourcenpfad `/streams` aus.
4. Geben Sie für Resource name (Ressourcenname) **`{stream-name}`** ein.

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

So konfigurieren und testen Sie die GET-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource/{stream-name} und wählen Sie dann Methode erstellen aus.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`DescribeStream`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

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

    Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode `GET /streams` einzurichten.
14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung `GET /streams/{stream-name}` der Integrationsanforderung `POST /?Action=DescribeStream` zuzuweisen:

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

Diese Zuweisungsvorlage generiert die erforderliche Nutzlast der Integrationsanforderung für die Aktion `DescribeStream` von Kinesis aufgrund des Pfadparameterwerts `stream-name` der Methodenanforderung.

15. Klicken Sie auf die Registerkarte Test, um die `GET /stream/{stream-name}`-Methode zum Aufrufen der `DescribeStream`-Aktion in Kinesis zu testen.
16. Geben Sie unter Stream-Name den Namen eines vorhandenen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" mit einer Nutzlast ähnlich der folgenden zurückgegeben:


```
{
  "StreamDescription": {
    "HasMoreShards": false,
    "RetentionPeriodHours": 24,
    "Shards": [
      {
        "HashKeyRange": {
          "EndingHashKey": "680564733841876926926749214863536422 90",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461454070523309915164834022007924120923395850242"
        },
        "ShardId": "shardId-000000000000"
      },
      ...
      {
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768211455",
          "StartingHashKey": "272225893536750770770699685945414569164"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461543273504104037657400164881014714369419771970"
        },
        "ShardId": "shardId-000000000004"
      }
    ],
    "StreamARN": "arn:aws:kinesis:us-east-1:12345678901:stream/myStream",
    "StreamName": "myStream",
```

```
    "StreamStatus": "ACTIVE"
  }
}
```

Nach der Bereitstellung der API können Sie eine REST-Anforderung für diese API-Methode erstellen:

```
GET https://your-api-id.execute-api.region.amazonaws.com/stage/streams/myStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z
```

So konfigurieren und testen Sie die POST-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource/{stream-name} und dann Methode erstellen aus.
2. Wählen Sie unter Method type (Methodentyp) die Option POST aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`CreateStream`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

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

Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode `GET /streams` einzurichten.

14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung `POST /streams/{stream-name}` der Integrationsanforderung `POST /?Action=CreateStream` zuzuweisen:

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

In der vorhergehenden Zuweisungsvorlage legen wir `ShardCount` auf einen festen Wert von 5 fest, wenn vom Client kein Wert in der Nutzlast der Methodenanforderung angegeben wird.

15. Klicken Sie auf die Registerkarte Test, um die `POST /stream/{stream-name}`-Methode zum Aufrufen der `CreateStream`-Aktion in Kinesis zu testen.
16. Geben Sie für Path unter Stream-Name den Namen eines neuen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" ohne Daten zurückgegeben.

Nach der Bereitstellung der API können Sie auch eine REST-API-Anfrage im Hinblick auf die POST-Methode für eine Stream-Ressource erstellen, um die `CreateStream`-Aktion in Amazon Kinesis aufzurufen:

```
POST https://your-api-id.execute-api.region.amazonaws.com/stage/streams/yourStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{
    "ShardCount": 5
}
```

Konfigurieren und Testen der DELETE-Methode für eine Stream-Ressource

1. Wählen Sie die Ressource/{stream-name} und dann Methode erstellen aus.
2. Wählen Sie als Methodentyp LÖSCHEN aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`DeleteStream`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

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

    Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode `GET /streams` einzurichten.
14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung `DELETE /streams/{stream-name}` der entsprechenden Integrationsanforderung `POST /?Action=DeleteStream` zuzuweisen:

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

    Diese Zuweisungsvorlage generiert die erforderliche Eingabe für die Aktion `DELETE /streams/{stream-name}` aus dem vom Client bereitgestellten URL-Pfad `stream-name`.

15. Klicken Sie auf die Registerkarte Test, um die `DELETE /stream/{stream-name}`-Methode zum Aufrufen der `DeleteStream`-Aktion in Kinesis zu testen.
16. Geben Sie unter Stream-Name den Namen eines vorhandenen Kinesis-Streams für Pfad ein.
17. Wählen Sie Test aus. Wenn der Test erfolgreich ist, wird die Antwort "200 OK" ohne Daten zurückgegeben.

    Nach der Bereitstellung der API können Sie auch die folgende REST-API-Anfrage im Hinblick auf die DELETE-Methode für die Stream-Ressource erstellen, um die `DeleteStream`-Aktion in Amazon Kinesis aufzurufen:

```
DELETE https://your-api-id.execute-api.region.amazonaws.com/stage/
streams/yourStream HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{}
```

Datensätze aus einem Stream in Kinesis abrufen und Datensätze zu einem Stream hinzufügen

Nachdem Sie einen Stream in Kinesis erstellt haben, können Sie dem Stream Datensätze hinzufügen und die Daten aus dem Stream lesen. Zum Hinzufügen von Datensätzen muss die PutRecordAktion PutRecordsoder in Kinesis aufgerufen werden. Erstere fügt dem Stream mehrere Datensätze hinzu, letztere hingegen einen einzigen Datensatz.

```
POST /?Action=PutRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "Records": [
```

```
        {
            "Data": blob,
            "ExplicitHashKey": "string",
            "PartitionKey": "string"
        }
    ],
    "StreamName": "string"
}
```

oder

```
POST /?Action=PutRecord HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "Data": blob,
    "ExplicitHashKey": "string",
    "PartitionKey": "string",
    "SequenceNumberForOrdering": "string",
    "StreamName": "string"
}
```

Hier identifiziert `StreamName` den Zieldatenstrom zum Hinzufügen von Datensätzen. `StreamName`, `Data`, und `PartitionKey` sind erforderliche Eingabedaten. In diesem Beispiel werden die Standardwerte für alle optionalen Eingabedaten verwendet. Es erfolgt keine explizite Angabe entsprechender Werte in der Eingabe für die Methodenanforderung.

Das Lesen von Daten in Kinesis läuft darauf hinaus, die GetRecordsAktion aufzurufen:

```
POST /?Action=GetRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes
```

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

Hier wird der Quell-Stream, von dem die Datensätze abgerufen werden, im erforderlichen `ShardIterator`-Wert angegeben. Dies wird in der folgenden Kinesis-Aktion zum Abrufen eines gemeinsamen Iterators verdeutlicht:

```
POST /?Action=GetShardIterator HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardId": "string",
    "ShardIteratorType": "string",
    "StartingSequenceNumber": "string",
    "StreamName": "string"
}
```

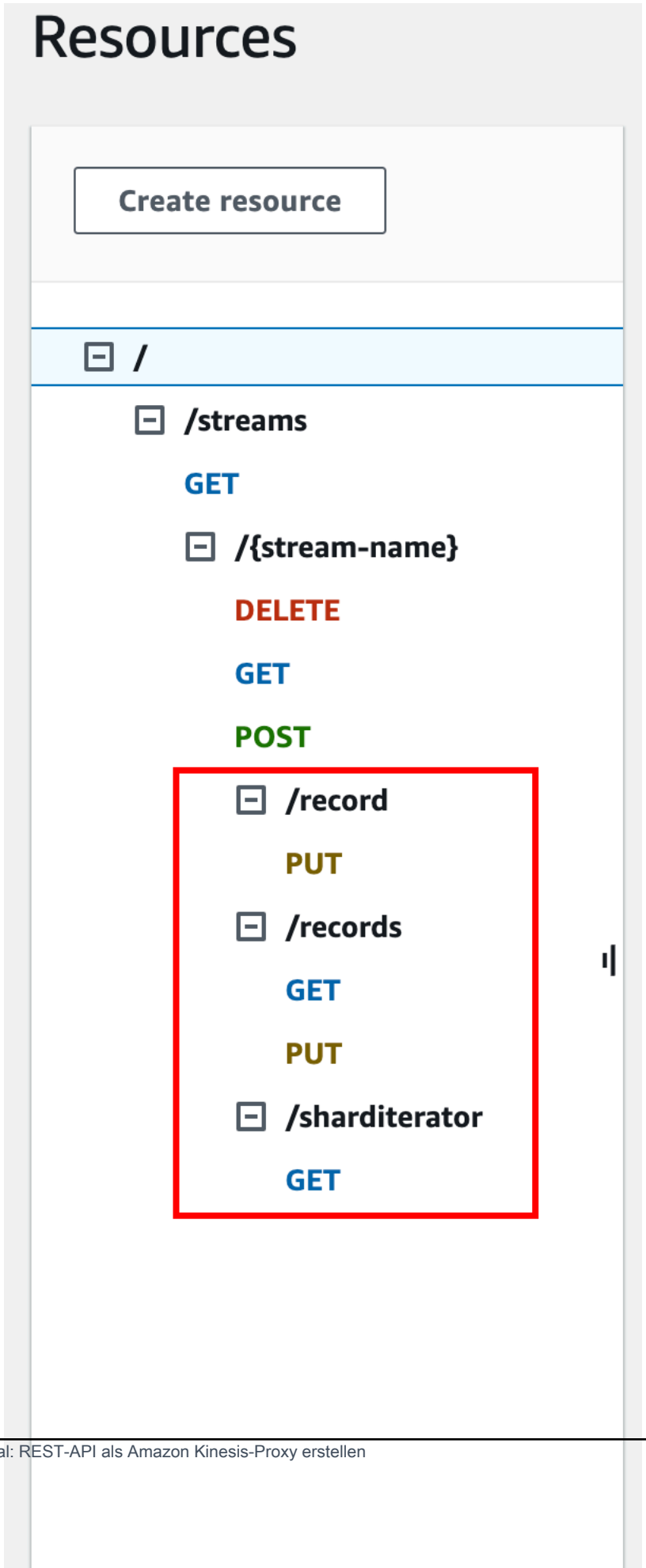
Für die Aktionen `GetRecords` und `PutRecords` werden die Methoden `GET` und `PUT` entsprechend für eine `/records`-Ressource bereitgestellt, die einem benannten Stream hinzugefügt wird (`/{stream-name}`). Ebenso wird die `PutRecord`-Aktion als `PUT`-Methode für eine `/record`-Ressource verfügbar gemacht.

Da die Aktion `GetRecords` als Eingabe einen `ShardIterator`-Wert verwendet, der durch Aufrufen der Hilfsaktion `GetShardIterator` gewonnen wurde, stellen Sie eine `GET`-Hilfsmethode für eine `ShardIterator`-Ressource bereit (`/sharditerator`).

So erstellen Sie die Ressourcen /record, /records und /sharditerator

1. Wählen Sie die Ressource/{stream-name} und dann Ressource erstellen aus.
2. Die Proxy-Ressource bleibt ausgeschaltet.
3. Wählen Sie für Ressourcenpfad `/{stream-name}` aus.

4. Geben Sie für Resource name (Ressourcenname) **`record`** ein.
5. CORS (Cross Origin Resource Sharing) bleibt ausgeschaltet.
6. Wählen Sie Create Resource (Ressource erstellen) aus.
7. Wiederholen Sie die Schritte oben, um eine /records- und eine /sharditerator-Ressource zu erstellen. Die endgültige API sollte wie folgt aussehen:


Resources
Create resource
/
/streams
GET
/{stream-name}
DELETE
GET
POST
/record
PUT
/records
GET
PUT
/sharditerator
GET

In den folgenden vier Verfahren wird beschrieben, wie Sie die jeweiligen Methoden einrichten, Daten aus den Methodenanforderungen den Integrationsanforderungen zuweisen und die Methoden testen.

Einrichten und Testen der **`PUT /streams/{stream-name}/record`**-Methode zum Aufruf von **`PutRecord`** in Kinesis:

1. Wählen Sie die Option /record und dann die Methode Create aus.
2. Wählen Sie unter Methodentyp PUT aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`PutRecord`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

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

    Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode `GET /streams` einzurichten.
14. Fügen Sie die folgende Textzuweisungsvorlage hinzu, um Daten der Methodenanforderung `PUT /streams/{stream-name}/record` der entsprechenden Integrationsanforderung `POST /?Action=PutRecord` zuzuweisen:

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

In dieser Zuweisungsvorlage wird davon ausgegangen, dass die Nutzlast der Methodenanforderung folgendes Format aufweist:

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

Diese Daten können mithilfe des folgenden JSON-Schemas modelliert werden:

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

Sie können ein Modell mit diesem Schema erstellen und es verwenden, um die Generierung der Zuweisungsvorlage zu vereinfachen. Sie können eine Zuweisungsvorlage jedoch auch ohne ein Modell erstellen.

15. Testen Sie die Methode `PUT /streams/{stream-name}/record`, indem Sie die Pfadvariable `stream-name` auf den Namen eines vorhandenen Streams festlegen, eine Nutzlast im erforderlichen Format bereitstellen und die Methodenanforderung dann senden. Das Ergebnis im Erfolgsfall ist die Antwort `200 OK` mit einer Nutzlast in folgendem Format:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-000000000004"
}
```

So richten Sie die **`PUT /streams/{stream-name}/records`**-Methode zum Aufruf von **`PutRecords`** in Kinesis ein und testen sie:

1. Wählen Sie die Ressource /records und anschließend die Methode Create aus.
2. Wählen Sie unter Methodentyp PUT aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`PutRecords`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie Methode erstellen aus.
13. Fügen Sie im Abschnitt Integrationsanfrage die folgenden Parameter für URL-Anforderungsheader hinzu:

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

    Die Aufgabe folgt demselben Verfahren, um die Zuweisung des Anforderungsparameters für die Methode `GET /streams` einzurichten.
14. Fügen Sie die folgende Zuweisungsvorlage hinzu, um Daten der `PUT /streams/{stream-name}/records`-Methodenanforderung der entsprechenden `POST /?Action=PutRecords`-Integrationsanforderung zuzuweisen:

    ```
    {
        "StreamName": "$input.params('stream-name')",
        "Records": [
           #foreach($elem in $input.path('$.records'))
              {
                "Data": "$util.base64Encode($elem.data)",
                "PartitionKey": "$elem.partition-key"
              }#if($foreach.hasNext),#end
            #end
    ```

```
    ]
}
```

In dieser Zuweisungsvorlage wird davon ausgegangen, dass die Nutzlast der Methodenanforderung anhand des folgenden JSON-Schemas modelliert werden kann:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecords proxy payload data",
  "type": "object",
  "properties": {
    "records": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "data": { "type": "string" },
          "partition-key": { "type": "string" }
        }
      }
    }
  }
}
```

Sie können ein Modell mit diesem Schema erstellen und es verwenden, um die Generierung der Zuweisungsvorlage zu vereinfachen. Sie können eine Zuweisungsvorlage jedoch auch ohne ein Modell erstellen.

In diesem Tutorial haben wir zwei geringfügig unterschiedliche Nutzdatenformate verwendet, um zu veranschaulichen, dass ein API-Entwickler je nach Bedarf das Backend-Datenformat für den Client offenlegen oder gegenüber dem Client verbergen kann. Eines dieser Formate ist für die Methode `PUT /streams/{stream-name}/records` bestimmt (oben). Ein anderes Format wird für die Methode `PUT /streams/{stream-name}/record` (im vorherigen Verfahren) verwendet. In einer Produktionsumgebung sollten beide Formate konsistent bleiben.

15. Testen Sie die Methode `PUT /streams/{stream-name}/records`, indem Sie die Pfadvariable `stream-name` auf einen vorhandenen Stream festlegen, die folgende Nutzlast bereitstellen und die Methodenanforderung senden.

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

Das Ergebnis im Erfolgsfall ist die Antwort "200 OK" mit einer Nutzlast, die der folgenden Ausgabe ähnelt:

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

Richten Sie die **`GET /streams/{stream-name}/sharditerator`**-Methode ein und testen Sie sie, indem Sie **`GetShardIterator`** in Kinesis aufrufen

Die Methode `GET /streams/{stream-name}/sharditerator` ist eine Hilfsmethode, um einen erforderlichen gemeinsamen Iterator vor dem Aufrufen der Methode `GET /streams/{stream-name}/records` zu beschaffen.

1. Wählen Sie die Ressource /sharditerator und anschließend Methode erstellen aus.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.

3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`GetShardIterator`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen.

    Die `GetShardIterator` Aktion erfordert die Eingabe eines ShardId Werts. Zur Weitergabe eines vom Client bereitgestellten `ShardId`-Werts wird der Methodenanforderung ein `shard-id`-Abfrageparameter wie nachfolgend dargestellt hinzugefügt:
13. Wählen Sie Abfragezeichenfolge hinzufügen aus.
14. Geben Sie unter Name **`shard-id`** ein.
15. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
16. Wählen Sie Methode erstellen aus.
17. Fügen Sie im Abschnitt Integrationsanforderung die folgende Zuweisungsvorlage hinzu, um die erforderlichen Eingaben (`ShardId` und `StreamName`) für die `GetShardIterator`-Aktion aus den Parametern `shard-id` und `stream-name` der Methodenanforderung zu generieren. Durch die Zuweisungsvorlage wird außerdem `ShardIteratorType` auf die Standardeinstellung `TRIM_HORIZON` festgelegt.

    ```
    {
        "ShardId": "$input.params('shard-id')",
        "ShardIteratorType": "TRIM_HORIZON",
        "StreamName": "$input.params('stream-name')"
    }
    ```
18. Verwenden Sie die Option Test in der Amazon API Gateway-Konsole, geben Sie einen vorhandenen Stream-Namen als Wert der Variablen `stream-name` Path ein, setzen Sie die `shard-id` Query string auf einen vorhandenen `ShardId`-Wert (z. B. `shard-000000000004`) und wählen Sie Test.

Im Erfolgsfall sieht die Ausgabe der Antwortnutzlast folgendermaßen oder ähnlich aus:

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

Notieren Sie sich den Wert für `ShardIterator`. Sie benötigen ihn, um Datensätze aus einem Stream abzurufen.

So konfigurieren und testen Sie die **`GET /streams/{stream-name}/records`**-Methode zum Aufrufen der **`GetRecords`**-Aktion in Kinesis:

1. Wählen Sie die Ressource /records und anschließend die Methode Create aus.
2. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.
3. Als Integrationstyp wählen Sie AWS -Dienst aus.
4. Wählen Sie für den AWS-Region Ort aus AWS-Region, an dem Sie Ihren Kinesis-Stream erstellt haben.
5. Für AWS-Service wählen Sie Kinesis aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie in HTTP method POST.
8. Für Aktionstyp wählen Sie Aktionsname verwenden aus.
9. Für Aktionsname geben Sie **`GetRecords`** ein.
10. Für Ausführungsrolle geben Sie den ARN Ihrer Ausführungsrolle ein.
11. Die Voreinstellung unter Passthrough für Content-Handling bleibt unverändert.
12. Wählen Sie HTTP-Anforderungsheader aus.

    Die `GetRecords`-Aktion erfordert die Eingabe eines `ShardIterator`-Werts. Um einen vom Kunden bereitgestellten `ShardIterator` Wert zu übergeben, fügen wir der Methodenanforderung einen `Shard-Iterator` Header-Parameter hinzu.
13. Wählen Sie Add header.
14. Geben Sie unter Name **`Shard-Iterator`** ein.
15. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.
16. Wählen Sie Methode erstellen aus.

17. Fügen Sie die folgende Textzuweisungsvorlage im Abschnitt Integrationsanforderung hinzu, um den `Shard-Iterator`-Header-Parameterwert dem `ShardIterator`-Eigenschaftswert der JSON-Nutzlast für die `GetRecords`-Aktion in Kinesis zuzuweisen.

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

18. Mithilfe der Option Test in der Amazon-API-Gateway-Konsole geben Sie einen vorhandenen Stream-Namen als Wert der Variablen `stream-name`-Pfad ein, setzen den `Shard-Iterator`-Header auf den `ShardIterator`-Wert aus dem Testdurchlauf der `GET /streams/{stream-name}/sharditerator`-Methode (oben) und klicken dann auf Test.

    Im Erfolgsfall sieht die Ausgabe der Antwortnutzlast folgendermaßen oder ähnlich aus:

```
{
  "MillisBehindLatest": 0,
  "NextShardIterator": "AAAAAAAAAAF...",
  "Records": [ ... ]
}
```

OpenAPI-Definitionen einer Beispiel-API als Kinesis-Proxy

Nachfolgend finden Sie OpenAPI-Definitionen für die in diesem Tutorial verwendete Beispiel-API als Kinesis-Proxy.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "KinesisProxy",
    "version": "2016-03-31T18:25:32Z"
  },
  "paths": {
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
```

```
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "shard-id",
            "in": "query",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
```

```
        "httpMethod": "POST"
      }
    }
  },
  "/streams/{stream-name}/records": {
    "get": {
      "parameters": [
        {
          "name": "stream-name",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        },
        {
          "name": "Shard-Iterator",
          "in": "header",
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
        "responses": {
          "default": {
            "statusCode": "200"
          }
        },
```

```
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
              }
            },
            "application/x-amz-json-1.1": {
              "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
              }
            }
          },
          "required": true
```

```
        },
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n           \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n       \"Data
\" : \"$elem.data\",\n       \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "parameters": [
```

```
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
```

```
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
```

```
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          },
          "400": {
            "description": "400 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
          },
          "500": {
            "description": "500 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
```

```
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
```

```
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
```

```
    },
    "/streams": {
      "get": {
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "components": {
    "schemas": {
      "Empty": {
        "type": "object"
      },
      "PutRecordsMethodRequestPayload": {
        "type": "object",
```

```
        "properties": {
          "records": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "data": {
                  "type": "string"
                },
                "partition-key": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/streams": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "responses": {
```

```
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
```

```
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
```

```
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
```

```
            "$ref": "#/definitions/Empty"
          },
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        },
        "400": {
          "description": "400 response",
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        },
        "500": {
          "description": "500 response",
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
        "responses": {
          "4\\d{2}": {
            "statusCode": "400",
            "responseParameters": {
              "method.response.header.Content-Type":
"integration.response.header.Content-Type"
            }
          },
          "default": {
            "statusCode": "200",
            "responseParameters": {
              "method.response.header.Content-Type":
"integration.response.header.Content-Type"
            }
          },
```

```
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
```

```
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "required": false,
            "type": "string"
```

```
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "consumes": [
          "application/json",
          "application/x-amz-json-1.1"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "required": false,
```

```
            "type": "string"
          },
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          },
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
```

```
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n            \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n        \"Data
\" : \"$elem.data\",\n        \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "shard-id",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
```

```
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object"
    },
    "PutRecordsMethodRequestPayload": {
      "type": "object",
      "properties": {
        "records": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "data": {
                "type": "string"
              },
              "partition-key": {
```

```
                "type": "string"
              }
            }
          }
        }
      }
    }
  }
}
```

Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI

Das folgende Tutorial zeigt, wie Sie eine PetStore API erstellen, die die `GET /pets/{petId}` Methoden `GET /pets` und unterstützt. Die Methoden sind in einen HTTP-Endpunkt integriert. Sie können diesem Tutorial mit dem AWS SDK für JavaScript, dem SDK for Python (Boto3) oder dem folgen. AWS CLI Sie verwenden die folgenden Funktionen oder Befehle, um Ihre API einzurichten:

JavaScript v3

- CreateRestApiCommand
- CreateResourceCommand
- PutMethodCommand
- PutMethodResponseCommand
- PutIntegrationCommand
- PutIntegrationResponseCommand
- CreateDeploymentCommand

Python

- create_rest_api
- Ressource erstellen
- put_methode
- put_method_response
- put_integration

- Antwort auf put_integration_
- Bereitstellung erstellen

AWS CLI

- create-rest-api
- Ressource erstellen
- Put-Methode
- put-method-response
- Put-Integration
- put-integration-response
- Bereitstellung erstellen

Weitere Informationen zum AWS SDK für JavaScript Version 3 finden Sie unter Wozu dient das AWS SDK? JavaScript . Weitere Informationen zum SDK for Python (Boto3) finden Sie unter. AWS SDK for Python (Boto3) Weitere Informationen zu dem finden Sie AWS CLI unter Was ist der? AWS CLI.

Richten Sie eine Edge-optimierte API PetStore ein

In diesem Tutorial verwenden Beispielbefehle Platzhalterwerte für Wert-IDs wie API-ID und Ressourcen-ID. Wenn Sie das Tutorial abgeschlossen haben, ersetzen Sie diese Werte durch Ihre eigenen.

So richten Sie mithilfe von SDKs eine Edge-optimierte PetStore API ein AWS

1. Verwenden Sie das folgende Beispiel, um eine Entität zu erstellen: `RestApi`

   JavaScript v3

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateRestApiCommand({
    name: "Simple PetStore (JavaScript v3 SDK)",
    description: "Demo API created using the AWS SDK for JavaScript v3",
    version: "0.00.001",
    binaryMediaTypes: [
```

```
    '*']
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.error(Couldn't create API:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt Ihre API-ID und die Root-Ressourcen-ID Ihrer API in einer Ausgabe wie der folgenden zurück:

```
{
  id: 'abc1234',
  name: 'PetStore (JavaScript v3 SDK)',
  description: 'Demo API created using the AWS SDK for node.js',
  createdDate: 2017-09-05T19:32:35.000Z,
  version: '0.00.001',
  rootResourceId: 'efg567'
  binaryMediaTypes: [ '*' ]
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_rest_api(
        name='Simple PetStore (Python SDK)',
        description='Demo API created using the AWS SDK for Python',
        version='0.00.001',
        binaryMediaTypes=[
            '*'
        ]
    )
except botocore.exceptions.ClientError as error:
```

```
    logger.exception("Couldn't create REST API %s.", error)
    raise
attribute=["id","name","description","createdDate","version","binaryMediaTypes","apiKeyS
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Ihre API-ID und die Root-Ressourcen-ID Ihrer API in einer Ausgabe wie der folgenden zurück:

```
{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
 'Demo API created using the AWS SDK for Python', 'createdDate':
 datetime.datetime(2024, 4, 3, 14, 31, 39, tzinfo=tzlocal()), 'version':
 '0.00.001', 'binaryMediaTypes': ['*'], 'apiKeySource': 'HEADER',
 'endpointConfiguration': {'types': ['EDGE']}, 'disableExecuteApiEndpoint':
 False, 'rootResourceId': 'efg567'}
```

AWS CLI

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-
west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "id": "abcd1234",
    "name": "Simple PetStore (AWS CLI)",
    "createdDate": "2022-12-15T08:07:04-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg567"
}
```

Die von Ihnen erstellte API hat die API-ID `abcd1234` und die Root-Ressourcen-ID von`efg567`. Sie verwenden diese Werte bei der Einrichtung Ihrer API.

2. Als Nächstes fügen Sie eine untergeordnete Ressource unter dem Stamm an und geben die `RootResourceId` als `parentId` Eigenschaftswert an. Verwenden Sie das folgende Beispiel, um eine `/pets` Ressource für Ihre API zu erstellen:

   JavaScript v3

   ```
   import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
   gateway";
   (async function (){
   const apig = new APIGatewayClient({region:"us-east-1"});
   const command = new CreateResourceCommand({
       restApiId: 'abcd1234',
       parentId: 'efg567',
       pathPart: 'pets'
   });
   try {
       const results = await apig.send(command)
       console.log(results)
   } catch (err) {
       console.log("The '/pets' resource setup failed:\n", err)
   }
   })();
   ```

   Ein erfolgreicher Aufruf gibt Informationen über Ihre Ressource in einer Ausgabe wie der folgenden zurück:

   ```
   {
       "path": "/pets",
       "pathPart": "pets",
       "id": "aaa111",
       "parentId": "efg567'"
   }
   ```

   Python

   ```
   import botocore
   import boto3
   import logging

   logger = logging.getLogger()
   apig = boto3.client('apigateway')
   ```

```
try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='efg567',
        pathPart='pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Informationen über Ihre Ressource in einer Ausgabe wie der folgenden zurück:

```
{'id': 'aaa111', 'parentId': 'efg567', 'pathPart': 'pets', 'path': '/pets'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id efg567 \
  --path-part pets
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "id": "aaa111",
    "parentId": "efg567",
    "pathPart": "pets",
    "path": "/pets"
}
```

Die `/pets` von Ihnen erstellte Ressource hat die Ressourcen-ID`aaa111`. Sie verwenden diesen Wert bei der Einrichtung Ihrer API.

3. Als Nächstes fügen Sie der Ressource eine untergeordnete `/pets` Ressource hinzu. Diese Ressource `/{petId}` hat einen Pfadparameter für den. Um einen `{petId}` Pfadteil zu einem

Pfadparameter zu machen, setzen Sie ihn in ein Paar geschweifter Klammern,. `{ }` Verwenden Sie das folgende Beispiel, um eine `/pets/{petId}` Ressource für Ihre API zu erstellen:

JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'aaa111',
    pathPart: '{petId}'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets/{petId}' resource setup failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt Informationen über Ihre Ressource in einer Ausgabe wie der folgenden zurück:

```
{
    "path": "/pets/{petId}",
    "pathPart": "{petId}",
    "id": "bbb222",
    "parentId": "aaa111'"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
```

```
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='aaa111',
        pathPart='{petId}'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets/{petId}' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt Informationen über Ihre Ressource in einer Ausgabe wie der folgenden zurück:

```
{'id': 'bbb222', 'parentId': 'aaa111', 'pathPart': '{petId}', 'path': '/pets/
{petId}'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id aaa111 \
  --path-part '{petId}'
```

Bei Erfolg gibt dieser Befehl folgende Antwort zurück:

```
{
    "id": "bbb222",
    "parentId": "aaa111",
    "path": "/pets/{petId}",
    "pathPart": "{petId}"
}
```

Die `/pets/{petId}` von Ihnen erstellte Ressource hat die Ressourcen-ID`bbb222`. Sie verwenden diesen Wert bei der Einrichtung Ihrer API.

4. In den folgenden zwei Schritten fügen Sie HTTP-Methoden zu Ihren Ressourcen hinzu. In diesem Tutorial stellen Sie die Methoden so ein, dass sie frei zugänglich sind, indem Sie die Option `authorization-type` auf setzen`NONE`. Damit nur authentifizierte Benutzer die

Methode aufrufen können, können Sie IAM-Rollen und -Richtlinien, einen Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bekannt) oder einen Amazon Cognito-Benutzerpool verwenden. Weitere Informationen finden Sie unter the section called “Zugriffskontrolle”.

Im folgenden Beispiel wird die `GET` HTTP-Methode zur `/pets` Ressource hinzugefügt:

JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    authorizationType: 'NONE'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets' method setup failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')
```

```
try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        authorizationType='NONE'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id aaa111 \
  --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2
```

Die erfolgreiche Ausgabe dieses Befehls ist wie folgt:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

5. Das folgende Beispiel fügt der `/pets/{petId}` Ressource die `GET` HTTP-Methode hinzu und legt die `requestParameters` Eigenschaft so fest, dass der vom Client bereitgestellte `petId` Wert an das Backend übergeben wird:

JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    authorizationType: 'NONE'
    requestParameters: {
        "method.request.path.petId" : true
    }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets/{petId}' method setup failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
       "method.request.path.petId": true
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
```

```
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        authorizationType='NONE',
        requestParameters={
            "method.request.path.petId": True
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets/{petId}' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired",
 "requestParameters" ]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False,
 'requestParameters': {'method.request.path.petId': True}}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2 \
  --request-parameters method.request.path.petId=true
```

Die erfolgreiche Ausgabe dieses Befehls ist wie folgt:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.petId": true
    }
```

```
}
```

6. Verwenden Sie das folgende Beispiel, um die Methodenantwort 200 OK für die `GET /pets` Methode hinzuzufügen:

   JavaScript v3

   ```
   import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
   gateway";
   (async function (){
   const apig = new APIGatewayClient({region:"us-east-1"});
   const command = new PutMethodResponseCommand({
       restApiId: 'abcd1234',
       resourceId: 'aaa111',
       httpMethod: 'GET',
       statusCode: '200'
   });
   try {
       const results = await apig.send(command)
       console.log(results)
   } catch (err) {
       console.log("Set up the 200 OK response for the 'GET /pets' method failed:
   \n", err)
   }
   })();
   ```

   Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

   ```
   {
       "statusCode": "200"
   }
   ```

   Python

   ```
   import botocore
   import boto3
   import logging

   logger = logging.getLogger()
   apig = boto3.client('apigateway')

   try:
   ```

```
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets' method
 failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "statusCode": "200"
}
```

7. Verwenden Sie das folgende Beispiel, um die Methodenantwort 200 OK für die `GET /pets/{petId}` Methode hinzuzufügen:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodResponseCommand({
    restApiId: 'abcd1234',
```

```
    resourceId: 'bbb222',
    httpMethod: 'GET',
    statusCode: '200'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the 200 OK response for the 'GET /pets/{petId}' method
 failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets/{petId}'
 method failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --status-code 200  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "statusCode": "200"
}
```

8. Im folgenden Beispiel wird eine Integration für die `GET /pets` Methode mit einem HTTP-Endpunkt konfiguriert. Der HTTP-Endpunkt ist`http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

   JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets' method of the API
 failed:\n", err)
}
```

```
})();
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "cacheNamespace": "ccc333"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /' method of the API
 failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-demo-
endpoint.execute-api.com/petstore/pets', 'cacheNamespace': 'ccc333'}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets' \
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "connectionType": "INTERNET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6sxz2j",
    "cacheKeyParameters": []
}
```

9. Im folgenden Beispiel wird eine Integration für die `GET /pets/{petId}` Methode mit einem HTTP-Endpunkt konfiguriert. Der HTTP-Endpunkt ist`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`. In diesem Schritt ordnen Sie den Pfadparameter dem Pfadparameter des Integrationsendpunkts von `petId` zu`id`.

   JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
```

```
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}'
    requestParameters: {
        "integration.request.path.id": "method.request.path.petId"
     }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets/{petId}' method of the
 API failed:\n", err)
}
})();
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "cacheNamespace": "ddd444",
    "requestParameters": {
       "integration.request.path.id": "method.request.path.petId"
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='ieps9b05sf',
```

```
        resourceId='t8zeb4',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}',
        requestParameters={
            "integration.request.path.id": "method.request.path.petId"
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /pets/{petId}' method
 of the API failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace", "requestParameters"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-
demo-endpoint.execute-api.com/petstore/pets/{id}', 'cacheNamespace':
 'ddd444', 'requestParameters': {'integration.request.path.id':
 'method.request.path.petId'}}}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}' \
  --request-parameters
 '{"integration.request.path.id":"method.request.path.petId"}' \
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
```

```
    "connectionType": "INTERNET",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    },
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "rjkmth",
    "cacheKeyParameters": []
}
```

10. Im folgenden Beispiel wird die Integrationsantwort für die `GET /pets` Integration hinzugefügt:

    JavaScript v3

    ```
    import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
    client-api-gateway";
    (async function (){
    const apig = new APIGatewayClient({region:"us-east-1"});
    const command = new PutIntegrationResponseCommand({
        restApiId: 'abcd1234',
        resourceId: 'aaa111',
        httpMethod: 'GET',
        statusCode: '200',
        selectionPattern: ''
    });
    try {
        const results = await apig.send(command)
        console.log(results)
    } catch (err) {
        console.log("The 'GET /pets' method integration response setup failed:\n",
     err)
    }
    })();
    ```

    Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

    ```
    {
        "selectionPattern": "",
        "statusCode": "200"
    }
    ```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets' method
 of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200 --selection-pattern ""  \
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "statusCode": "200",
    "selectionPattern": ""
```

```
}
```

11. Im folgenden Beispiel wird die Integrationsantwort für die `GET /pets/{petId}` Integration hinzugefügt:

    JavaScript v3

    ```
    import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
    client-api-gateway";
    (async function (){
    const apig = new APIGatewayClient({region:"us-east-1"});
    const command = new PutIntegrationResponseCommand({
        restApiId: 'abcd1234',
        resourceId: 'bbb222',
        httpMethod: 'GET',
        statusCode: '200',
        selectionPattern: ''
    });
    try {
        const results = await apig.send(command)
        console.log(results)
    } catch (err) {
        console.log("The 'GET /pets/{petId}' method integration response setup
     failed:\n", err)
    }
    })();
    ```

    Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

    ```
    {
        "selectionPattern": "",
        "statusCode": "200"
    }
    ```

    Python

    ```
    import botocore
    import boto3
    import logging

    logger = logging.getLogger()
    apig = boto3.client('apigateway')
    ```

```
try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets/{petId}'
 method of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Ein erfolgreicher Aufruf gibt die folgende Ausgabe zurück:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET
  --status-code 200 --selection-pattern ""
  --region us-west-2
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

Nachdem Sie die Integrationsantwort erstellt haben, kann Ihre API verfügbare Haustiere auf der PetStore Website abfragen und einzelne Haustiere mit einer bestimmten Kennung anzeigen. Bevor Ihre API von Ihren Kunden aufgerufen werden kann, müssen Sie sie bereitstellen. Wir empfehlen, dass Sie Ihre API testen, bevor Sie sie bereitstellen.

12. Das folgende Beispiel testet die `GET /pets` Methode:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    pathWithQueryString: '/',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets' method failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        pathWithQueryString='/',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets' failed: %s", error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id aaa111 /
  --http-method GET /
  --path-with-query-string '/'
```

13. Im folgenden Beispiel wird die `GET /pets/{petId}` Methode mit einem Wert `petId` von 3 getestet:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    pathWithQueryString: '/pets/3',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets/{petId}' method failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
```

```
        resourceId='bbb222',
        httpMethod='GET',
        pathWithQueryString='/pets/3',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets/{petId}' failed: %s",
 error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id bbb222 /
  --http-method GET /
  --path-with-query-string '/pets/3'
```

Nachdem Sie Ihre API erfolgreich getestet haben, können Sie sie in einer Phase bereitstellen.

14. Im folgenden Beispiel wird Ihre API in einer Phase mit dem Namen `test` bereitgestellt. Wenn Sie Ihre API in einer Phase bereitstellen, können API-Aufrufer Ihre API aufrufen.

    JavaScript v3

```
import {APIGatewayClient,  CreateDeploymentCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateDeploymentCommand({
    restApiId: 'abcd1234',
    stageName: 'test',
    stageDescription: 'test deployment'
});
try {
    const results = await apig.send(command)
    console.log("Deploying API succeeded\n", results)
} catch (err) {
    console.log("Deploying API failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_deployment(
        restApiId='ieps9b05sf',
        stageName='test',
        stageDescription='my test stage',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Error deploying stage  %s.", error)
    raise
print('Deploying API succeeded')
print(result)
```

AWS CLI

```
aws apigateway create-deployment --rest-api-id abcd1234 \
  --region us-west-2 \
  --stage-name test \
  --stage-description 'Test stage' \
  --description 'First deployment'
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "id": "ab1c1d",
    "description": "First deployment",
    "createdDate": "2022-12-15T08:44:13-08:00"
}
```

Ihre API kann jetzt von Kunden aufgerufen werden. Sie können diese API testen, indem Sie die `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets` URL in einen Browser eingeben und sie durch die ID Ihrer API ersetzen`abcd1234`.

Weitere Beispiele zum Erstellen oder Aktualisieren einer API mithilfe von AWS SDKs oder dem AWS CLI finden Sie unter Aktionen für API Gateway mithilfe von AWS SDKs.

Automatisieren Sie die Einrichtung Ihrer API

Anstatt Ihre API zu erstellen step-by-step, können Sie die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie OpenAPI oder Terraform verwenden AWS CloudFormation, um Ihre API zu erstellen.

OpenAPI 3.0-Definition

Sie können eine OpenAPI-Definition in API Gateway importieren. Weitere Informationen finden Sie unter the section called “OpenAPI”.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Simple PetStore (OpenAPI)",
    "description" : "Demo API created using OpenAPI",
    "version" : "2024-05-24T20:39:34Z"
  },
  "servers" : [ {
    "url" : "{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "Prod"
      }
    }
  } ],
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
```

```
            "default" : {
              "statusCode" : "200"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    },
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.id" : "method.request.path.petId"
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    }
  },
  "components" : { }
```

```
}
```

AWS CloudFormation Vorlage

Informationen zum Bereitstellen Ihrer AWS CloudFormation Vorlage finden Sie unter Einen Stack auf der AWS CloudFormation Konsole erstellen.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: Simple PetStore (AWS CloudFormation)
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetIdResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !Ref PetsResource
      PathPart: '{petId}'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PetIdMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
```

```
      RestApiId: !Ref Api
      ResourceId: !Ref PetIdResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestParameters:
        method.request.path.petId: true
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}
        RequestParameters:
          integration.request.path.id: method.request.path.petId
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: Prod
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/Prod'
```

Terraform-Konfiguration

Weitere Informationen zu Terraform finden Sie unter Terraform.

```
provider "aws" {
  region = "us-east-1" # Update with your desired region
}
resource "aws_api_gateway_rest_api" "Api" {
  name        = "Simple PetStore (Terraform)"
  description = "Demo API created using Terraform"
}
resource "aws_api_gateway_resource" "petsResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_rest_api.Api.root_resource_id
    path_part = "pets"
}
```

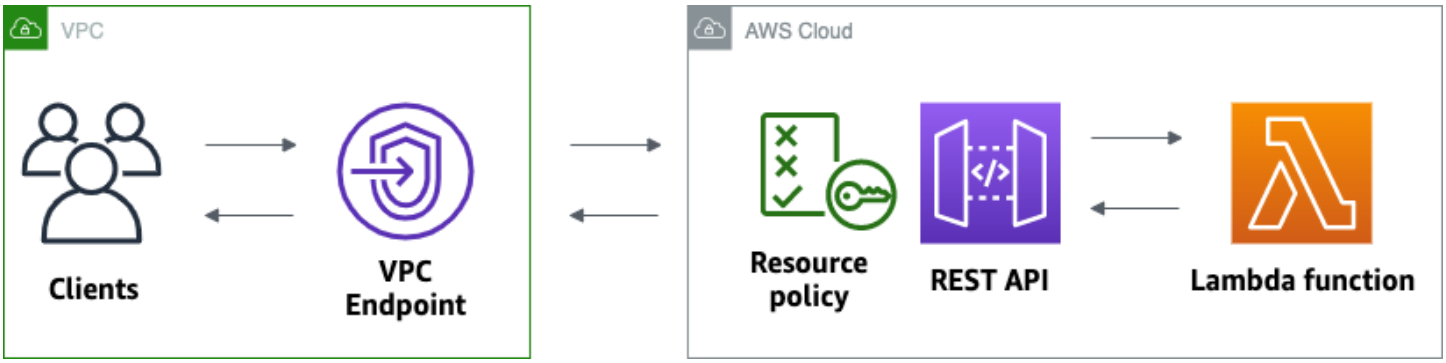
```
resource "aws_api_gateway_resource" "petIdResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_resource.petsResource.id
    path_part = "{petId}"
}
resource "aws_api_gateway_method" "petsMethodGet" {
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  resource_id   = aws_api_gateway_resource.petsResource.id
  http_method   = "GET"
  authorization = "NONE"
}


resource "aws_api_gateway_method_response" "petsMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code ="200"
}

resource "aws_api_gateway_integration" "petsIntegration" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  resource_id = aws_api_gateway_resource.petsResource.id
  http_method = aws_api_gateway_method.petsMethodGet.http_method
  type        = "HTTP"

  uri                     = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets"
  integration_http_method = "GET"
  depends_on              = [aws_api_gateway_method.petsMethodGet]
}

resource "aws_api_gateway_integration_response" "petsIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code = aws_api_gateway_method_response.petsMethodResponseGet.status_code
}

resource "aws_api_gateway_method" "petIdMethodGet" {
    rest_api_id   = aws_api_gateway_rest_api.Api.id
    resource_id   = aws_api_gateway_resource.petIdResource.id
    http_method   = "GET"
    authorization = "NONE"
```

```
    request_parameters = {"method.request.path.petId" = true}
}

resource "aws_api_gateway_method_response" "petIdMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code ="200"
}


resource "aws_api_gateway_integration" "petIdIntegration" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    type        = "HTTP"
    uri                     = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets/{id}"
    integration_http_method = "GET"
    request_parameters = {"integration.request.path.id" = "method.request.path.petId"}
    depends_on              = [aws_api_gateway_method.petIdMethodGet]
}

resource "aws_api_gateway_integration_response" "petIdIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code = aws_api_gateway_method_response.petIdMethodResponseGet.status_code
}


resource "aws_api_gateway_deployment" "Deployment" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  depends_on  =
 [aws_api_gateway_integration.petsIntegration,aws_api_gateway_integration.petIdIntegration ]
}
resource "aws_api_gateway_stage" "Stage" {
  stage_name    = "Prod"
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  deployment_id = aws_api_gateway_deployment.Deployment.id
}
```

Tutorial: Erstellen Sie eine private REST-API

In diesem Tutorial erstellen Sie eine private REST-API. Clients können nur von Ihrer Amazon VPC aus auf die API zugreifen. Die API ist vom öffentlichen Internet isoliert, was eine übliche Sicherheitsanforderung ist.

Dieses Tutorial nimmt ungefähr 30 Minuten in Anspruch. Zunächst verwenden Sie eine AWS CloudFormation Vorlage, um eine Amazon VPC, einen VPC-Endpunkt und eine AWS Lambda Funktion zu erstellen und eine Amazon EC2 EC2-Instance zu starten, mit der Sie Ihre API testen werden. Als Nächstes verwenden Sie die, AWS Management Console um eine private API zu erstellen und eine Ressourcenrichtlinie anzuhängen, die den Zugriff nur von Ihrem VPC-Endpunkt aus erlaubt. Zuletzt testen Sie Ihre API.



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

In diesem Tutorial wird AWS Management Console verwendet. Eine AWS CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter template.yaml.

Themen

Schritt 1: Erstellen von Abhängigkeiten

Laden Sie diese Vorlage herunter und entpacken Sie sie. AWS CloudFormation Sie verwenden die Vorlage, um alle Abhängigkeiten für Ihre private API zu erstellen, einschließlich einer Amazon VPC, eines VPC-Endpunkts und einer Lambda-Funktion, die als Backend Ihrer API dient. Sie erstellen die private API später.

Um einen Stapel zu erstellen AWS CloudFormation

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

AWS CloudFormation stellt die Abhängigkeiten für Ihre API bereit, was einige Minuten dauern kann. Wenn der Status Ihres AWS CloudFormation Stacks CREATE_COMPLETE lautet, wählen Sie Outputs. Notieren Sie sich Ihre VPC-Endpunkt-ID. Sie benötigen sie für spätere Schritte in diesem Tutorial.

Schritt 2: Erstellen einer privaten API

Sie erstellen eine private API, um nur Clients in Ihrer VPC den Zugriff darauf zu ermöglichen.

Erstellen einer privaten API

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie API erstellen und dann für Rest-API die Option Erstellen aus.

3. Geben Sie unter API name (API-Name) **`private-api-tutorial`** ein.
4. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
5. Geben Sie für VPC-Endpunkt-IDs die VPC-Endpunkt-ID aus den Ausgaben Ihres AWS CloudFormation Stacks ein.
6. Wählen Sie Create API (API erstellen) aus.

Schritt 3: Erstellen Sie eine Methode und Integration

Sie erstellen eine `GET` Methode und eine Lambda-Integration, um `GET`-Anfragen an Ihre API zu bearbeiten. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion, die Sie in Schritt 1 erstellt haben und gibt dann eine Antwort an den Client zurück.

So erstellen Sie eine Methode und Integration

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wählen Sie die /-Ressource aus und klicken Sie dann auf Methode erstellen.
4. Für Methodentyp wählen Sie `GET` aus.
5. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
6. Schalten Sie die Lambda-Proxy-Integration ein. Mit einer Lambda-Proxy-Integration sendet API Gateway ein Ereignis mit einer definierten Struktur an Lambda und transformiert die Antwort von Ihrer Lambda-Funktion in eine HTTP-Antwort.
7. Wählen Sie für die Lambda-Funktion die Funktion aus, die Sie mit der AWS CloudFormation Vorlage in Schritt 1 erstellt haben. Der Name der Funktion beginnt mit **`private-api-tutorial`**.
8. Wählen Sie Methode erstellen aus.

Schritt 4: Anhängen einer Ressourcenrichtlinie

Sie hängen Ihrer API eine Ressourcenrichtlinie an, die es Clients ermöglicht, Ihre API nur über Ihren VPC-Endpunkt aufzurufen. Um den Zugriff auf Ihre API weiter einzuschränken, können Sie auch eine VPC-Endpunktrichtlinie für Ihren VPC-Endpunkt konfigurieren, aber das ist für dieses Tutorial nicht erforderlich.

So hängen Sie eine Ressourcenrichtlinie an

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Ressourcenrichtlinie aus und klicken Sie dann auf Richtlinie erstellen.
4. Geben Sie die folgende Richtlinie ein: Ersetzen Sie *vpceID* durch Ihre VPC-Endpunkt-ID aus den Ausgaben Ihres Stacks. AWS CloudFormation

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpceID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*"
        }
    ]
}
```

5. Wählen Sie Änderungen speichern aus.

Schritt 5: Bereitstellen Ihrer API

Als Nächstes stellen Sie Ihre API bereit, um sie Clients in Ihrer Amazon VPC zur Verfügung zu stellen.

So stellen Sie eine API bereit

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Klicken Sie auf Deploy API.
4. Wählen Sie für Stufe die Option Neue Stufe aus.
5. Geben Sie für Stage name (Stufenname) **`test`** ein.
6. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
7. Wählen Sie Deploy (Bereitstellen) aus.

Jetzt sind Sie bereit, Ihre API zu testen.

Schritt 6: Stellen Sie sicher, dass Ihre API nicht öffentlich zugänglich ist

Verwenden Sie, `curl` um zu überprüfen, ob Sie Ihre API nicht außerhalb Ihrer Amazon VPC aufrufen können.

So testen Sie Ihre API

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wählen Sie im Hauptnavigationsbereich Stufen aus und klicken Sie dann auf die Stufe Test.
4. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren. Die URL sieht folgendermaßen aus `https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. Für den VPC-Endpunkt, den Sie in Schritt 1 erstellt haben, ist privates DNS aktiviert, sodass Sie die bereitgestellte URL zum Aufrufen Ihrer API verwenden können.
5. Verwenden Sie curl, um zu versuchen, Ihre API von außerhalb Ihrer VPC aufzurufen.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Curl zeigt an, dass der Endpunkt Ihrer API nicht aufgelöst werden kann. Wenn Sie eine andere Antwort erhalten, gehen Sie zurück zu Schritt 2 und stellen Sie sicher, dass Sie Privat für den Endpunkttyp Ihrer API auswählen.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/
test
```

Als Nächstes stellen Sie eine Verbindung zu einer Amazon-EC2-Instance in Ihrer VPC her, um Ihre API aufzurufen.

Schritt 7: Verbinden Sie sich mit einer Instance in Ihrer VPC und rufen Sie Ihre API auf

Als Nächstes testen Sie Ihre API aus Ihrer Amazon VPC heraus. Um auf Ihre private API zuzugreifen, stellen Sie eine Verbindung zu einer Amazon-EC2-Instance in Ihrer VPC her und verwenden dann curl, um Ihre API aufzurufen. Sie verwenden Systems-Manager Session-Manager, um eine Verbindung mit Ihrer Instance im Browser herzustellen.

So testen Sie Ihre API

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie Instances.
3. Wählen Sie den Instanznamen aus private-api-tutorial, den Sie mit der AWS CloudFormation Vorlage in Schritt 1 erstellt haben.
4. Wählen Sie Verbinden und wählen Sie dann Session-Manager.
5. Wählen Sie Verbinden , um eine browserbasierte Sitzung für Ihre Instance zu starten.
6. Verwenden Sie in Ihrer Session-Manager-Sitzung curl, um Ihre API aufzurufen. Sie können Ihre API aufrufen, da Sie eine Instance in Ihrer Amazon VPC verwenden.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Stellen Sie sicher, dass Sie die Antwort erhalte `Hello from Lambda!`.

Sie haben erfolgreich eine API erstellt, auf die nur von Ihrer Amazon VPC aus zugegriffen werden kann und dann überprüft, dass sie funktioniert.

Schritt 8: Bereinigen

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

So löschen Sie eine REST-API

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie auf der Seite APIs eine API aus. Wählen Sie API-Aktionen aus, klicken Sie auf API löschen und bestätigen Sie anschließend Ihre Auswahl.

Um einen AWS CloudFormation Stack zu löschen

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

Nächste Schritte: Automatisieren Sie mit AWS CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine vollständige AWS CloudFormation -Beispielvorlage finden Sie unter template.yaml.

Amazon API Gateway-HTTP-API-Tutorials

Die folgenden Tutorials bieten praktische Übungen, die Ihnen beim Kennenlernen von API Gateway HTTP APIs helfen.

Themen

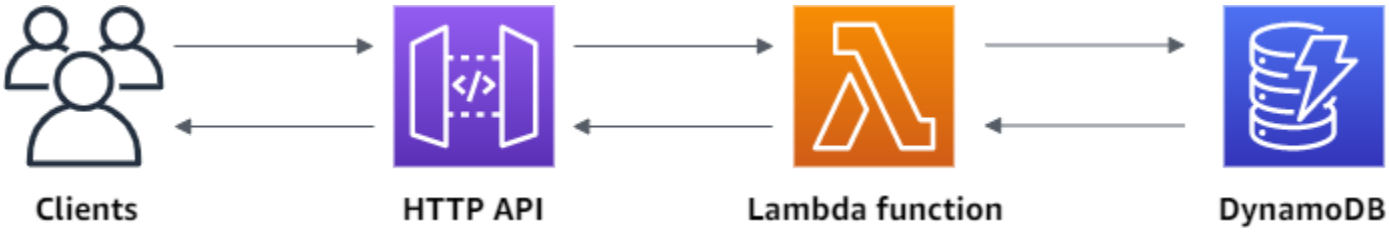
- Tutorial: Erstellen einer CRUD-API mit Lambda und DynamoDB
- Tutorial: Aufbau einer HTTP-API mit privater Integration in einen Amazon-ECS-Service

Tutorial: Erstellen einer CRUD-API mit Lambda und DynamoDB

In diesem Tutorial erstellen Sie eine serverlose API, die Elemente aus einer DynamoDB-Tabelle erstellt, liest, aktualisiert und löscht. DynamoDB ist ein vollständig verwalteter NoSQL-Datenbankservice, der schnelle und vorhersehbare Leistung nahtlos skalierbar bereitstellt. Dieses Tutorial dauert ungefähr 30 Minuten, und Sie können es im Rahmen des kostenlosen Kontingents von AWS durchgehen.

Zunächst erstellen Sie mit der DynamoDB-Konsole eine DynamoDB-Tabelle. Dann erstellen Sie mit der AWS Lambda Konsole eine Lambda-Funktion. Als Nächstes erstellen Sie eine HTTP-API mit der API-Gateway-Konsole. Zuletzt testen Sie Ihre API.

Wenn Sie Ihre HTTP-API aufrufen, leitet API Gateway die Anfrage an Ihre Lambda-Funktion weiter. Die Lambda-Funktion interagiert mit DynamoDB und gibt eine Antwort an API Gateway zurück. API Gateway gibt dann eine Antwort an Sie zurück.



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

In diesem Tutorial wird AWS Management Console verwendet. Eine AWS SAM Vorlage, mit der diese API und alle zugehörigen Ressourcen erstellt werden, finden Sie unter `template.yaml`.

Themen

- Schritt 1: Erstellen einer DynamoDB-Tabelle

Schritt 1: Erstellen einer DynamoDB-Tabelle

Sie verwenden eine DynamoDB-Tabelle, um Daten für Ihre API zu speichern.

Jedes Element hat eine eindeutige ID, die wir als Partitionsschlüssel für die Tabelle verwenden.

So erstellen Sie eine DynamoDB-Tabelle

1. Öffnen Sie die DynamoDB-Konsole unter https://console.aws.amazon.com/dynamodb/.
2. Wählen Sie Create table aus.
3. Geben Sie für Table name (Tabellenname) **`http-crud-tutorial-items`** ein.
4. Geben Sie für Partition key (Partitionsschlüssel) den Wert **`id`** ein.
5. Wählen Sie Create table (Tabelle erstellen) aus.

Schritt 2: Erstellen einer Lambda-Funktion

Sie erstellen eine Lambda-Funktion für das Backend Ihrer API. Diese Lambda-Funktion erstellt, liest, aktualisiert und löscht Elemente aus DynamoDB. Die Funktion verwendet Ereignisse von API Gateway, um zu bestimmen, wie mit DynamoDB interagiert wird. Der Einfachheit halber verwendet dieses Tutorial eine einzige Lambda-Funktion. Als Best Practice sollten Sie separate Funktionen für jede Route erstellen.

So erstellen Sie eine Lambda-Funktion:

1. Melden Sie sich bei der Lambda-Konsole unter https://console.aws.amazon.com/lambda an.
2. Wählen Sie Create function (Funktion erstellen).
3. Geben Sie für Function name (Funktionsname) **`http-crud-tutorial-function`** ein.

4. Wählen Sie für Laufzeit die neueste unterstützte Node.js- oder Python-Laufzeit aus.
5. Wählen Sie unter Berechtigungen die Option Standardeausführungsrolle ändern aus.
6. Wählen Sie Neue Rolle aus AWS Richtlinienvorlagen erstellen aus.
7. Geben Sie für Role name (Rollenname) den Namen **`http-crud-tutorial-role`** ein.
8. Wählen Sie für Policy templates (Richtlinienvorlagen) **`Simple microservice permissions`** aus. Diese Richtlinie gewährt der Lambda-Funktion die Berechtigung, mit DynamoDB zu interagieren.

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

9. Wählen Sie Create function (Funktion erstellen).
10. Öffnen Sie die Lambda-Funktion im Code-Editor der Konsole und ersetzen Sie ihren Inhalt durch den folgenden Code. Wählen Sie Deploy (Bereitstellen) aus, um Ihre Funktion zu aktualisieren.

Node.js

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "http-crud-tutorial-items";

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
```

```
  "Content-Type": "application/json",
};

try {
  switch (event.routeKey) {
    case "DELETE /items/{id}":
      await dynamo.send(
        new DeleteCommand({
          TableName: tableName,
          Key: {
            id: event.pathParameters.id,
          },
        })
      );
      body = `Deleted item ${event.pathParameters.id}`;
      break;
    case "GET /items/{id}":
      body = await dynamo.send(
        new GetCommand({
          TableName: tableName,
          Key: {
            id: event.pathParameters.id,
          },
        })
      );
      body = body.Item;
      break;
    case "GET /items":
      body = await dynamo.send(
        new ScanCommand({ TableName: tableName })
      );
      body = body.Items;
      break;
    case "PUT /items":
      let requestJSON = JSON.parse(event.body);
      await dynamo.send(
        new PutCommand({
          TableName: tableName,
          Item: {
            id: requestJSON.id,
            price: requestJSON.price,
            name: requestJSON.name,
          },
        })
```

```
        );
        body = `Put item ${requestJSON.id}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

Python

```
import json
import boto3
from decimal import Decimal

client = boto3.client('dynamodb')
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table('http-crud-tutorial-items')
tableName = 'http-crud-tutorial-items'


def lambda_handler(event, context):
    print(event)
    body = {}
    statusCode = 200
    headers = {
        "Content-Type": "application/json"
    }

    try:
        if event['routeKey'] == "DELETE /items/{id}":
            table.delete_item(
```

```
                Key={'id': event['pathParameters']['id']})
            body = 'Deleted item ' + event['pathParameters']['id']
        elif event['routeKey'] == "GET /items/{id}":
            body = table.get_item(
                Key={'id': event['pathParameters']['id']})
            body = body["Item"]
            responseBody = [
                {'price': float(body['price']), 'id': body['id'], 'name':
body['name']}]
            body = responseBody
        elif event['routeKey'] == "GET /items":
            body = table.scan()
            body = body["Items"]
            print("ITEMS----")
            print(body)
            responseBody = []
            for items in body:
                responseItems = [
                    {'price': float(items['price']), 'id': items['id'], 'name':
items['name']}]
                responseBody.append(responseItems)
            body = responseBody
        elif event['routeKey'] == "PUT /items":
            requestJSON = json.loads(event['body'])
            table.put_item(
                Item={
                    'id': requestJSON['id'],
                    'price': Decimal(str(requestJSON['price'])),
                    'name': requestJSON['name']
                })
            body = 'Put item ' + requestJSON['id']
    except KeyError:
        statusCode = 400
        body = 'Unsupported route: ' + event['routeKey']
    body = json.dumps(body)
    res = {
        "statusCode": statusCode,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": body
    }
    return res
```

Schritt 3: HTTP-API erstellen

Die HTTP-API bietet einen HTTP-Endpunkt für Ihre Lambda-Funktion. In diesem Schritt erstellen Sie eine leere API. In den folgenden Schritten konfigurieren Sie Routen und Integrationen, um Ihre API und Ihre Lambda-Funktion zu verbinden.

So erstellen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) und dann für HTTP API die Option Build (Erstellen) aus.
3. Geben Sie unter API name (API-Name) **`http-crud-tutorial-api`** ein.
4. Wählen Sie Next.
5. Wählen Sie für Configure routes (Routen konfigurieren) die Option Next (Weiter) aus, um die Routenerstellung zu überspringen. Sie werden Routen später erstellen.
6. Überprüfen Sie die Phase, die API Gateway für Sie erstellt, und wählen Sie dann Next (Weiter) aus.
7. Wählen Sie Create aus.

Schritt 4: Routen erstellen

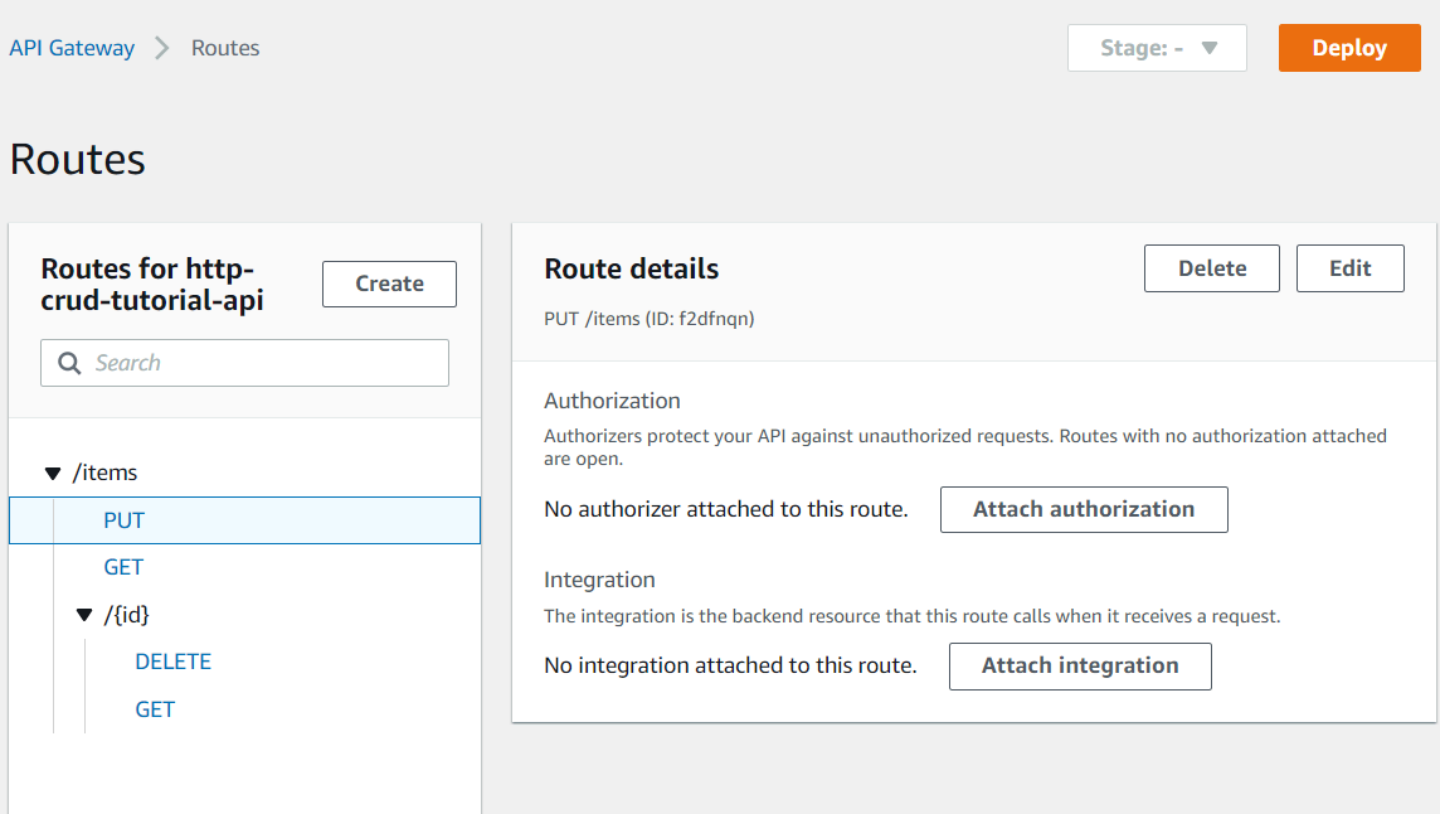
Routen sind eine Möglichkeit, eingehende API-Anforderungen an Backend-Ressourcen zu senden. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispie, `GET /items`. Für diese Beispiel-API erstellen wir vier Routen:

- `GET /items/{id}`
- `GET /items`
- `PUT /items`
- `DELETE /items/{id}`

So werden Routen erstellt

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.

3. Wählen Sie Routes (Routen) aus.
4. Wählen Sie Create aus.
5. Wählen Sie für Methode **`GET`** aus.
6. Für den Pfad geben Sie ei **`/items/{id}`**. `{id}` am Ende des Pfads ist ein Pfadparameter, den API Gateway vom Anforderungspfad abruft, wenn ein Client eine Anforderung stellt.
7. Wählen Sie Create aus.
8. Wiederholen Sie die Schritte 4-7 für `GET /items`, `DELETE /items/{id}` und `PUT /items`.



Schritt 5: Eine Integration erstellen

Sie erstellen eine Integration, um eine Route mit Backend-Ressourcen zu verbinden. Für diese Beispiel-API erstellen Sie eine Lambda-Integration, die Sie für alle Routen verwenden.

So wird eine Integration erstellt

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.

3. Wähle Sie Integrations (Integrationen) aus.
4. Wählen Sie Manage integrations (Integrationen verwalten) und dann Create (Erstellen) aus.
5. Überspringen Sie Attach this integration to a route (Diese Integration an eine Route anfügen). Dies werden Sie in einem späteren Schritt abschließen.
6. Für Integration type (Integrationstyp) wählen Sie Lambda Function (Lambda-Funktion) aus.
7. Geben Sie für die Lambda function (Lambda-Funktion) **`http-crud-tutorial-function`** ein.
8. Wählen Sie Create aus.

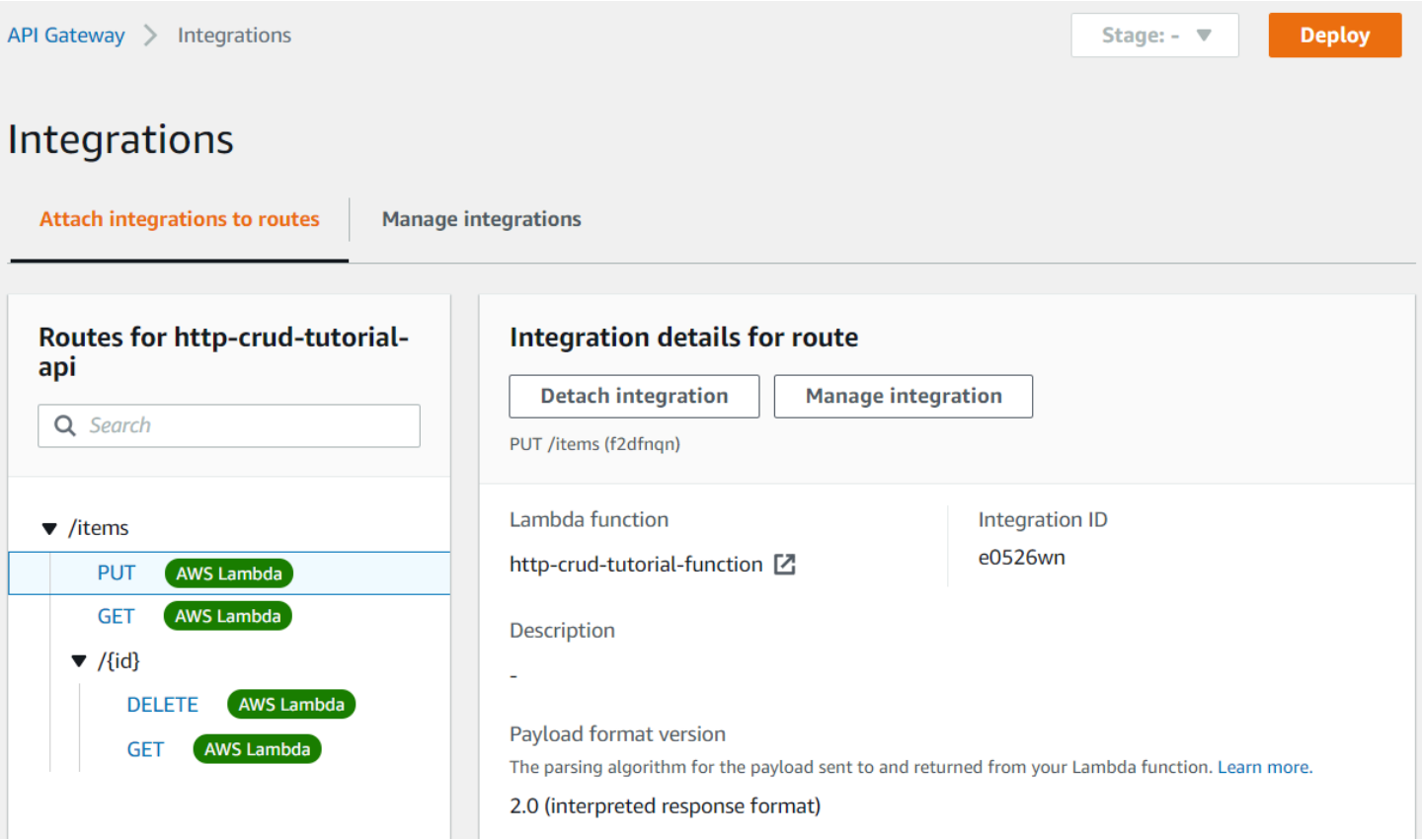
Schritt 6: Ihre Integration an Routen anfügen

Für diese Beispiel-API verwenden Sie dieselbe Lambda-Integration für alle Routen. Nachdem Sie die Integration allen Routen der API angefügt haben, wird Ihre Lambda-Funktion aufgerufen, wenn ein Client eine Ihrer Routen aufruft.

So werden Integrationen an Routen angefügt

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wähle Sie Integrations (Integrationen) aus.
4. Wählen Sie eine Route aus.
5. Wählen Sie unter Choose an existing integration (Eine vorhandene Integration auswählen) **`http-crud-tutorial-function`** aus.
6. Wählen Sie Attach integration (Integration anhängen) aus.
7. Wiederholen Sie die Schritte 4 bis 6 für alle Routen.

Alle Routen zeigen, dass eine AWS Lambda Integration angehängt ist.



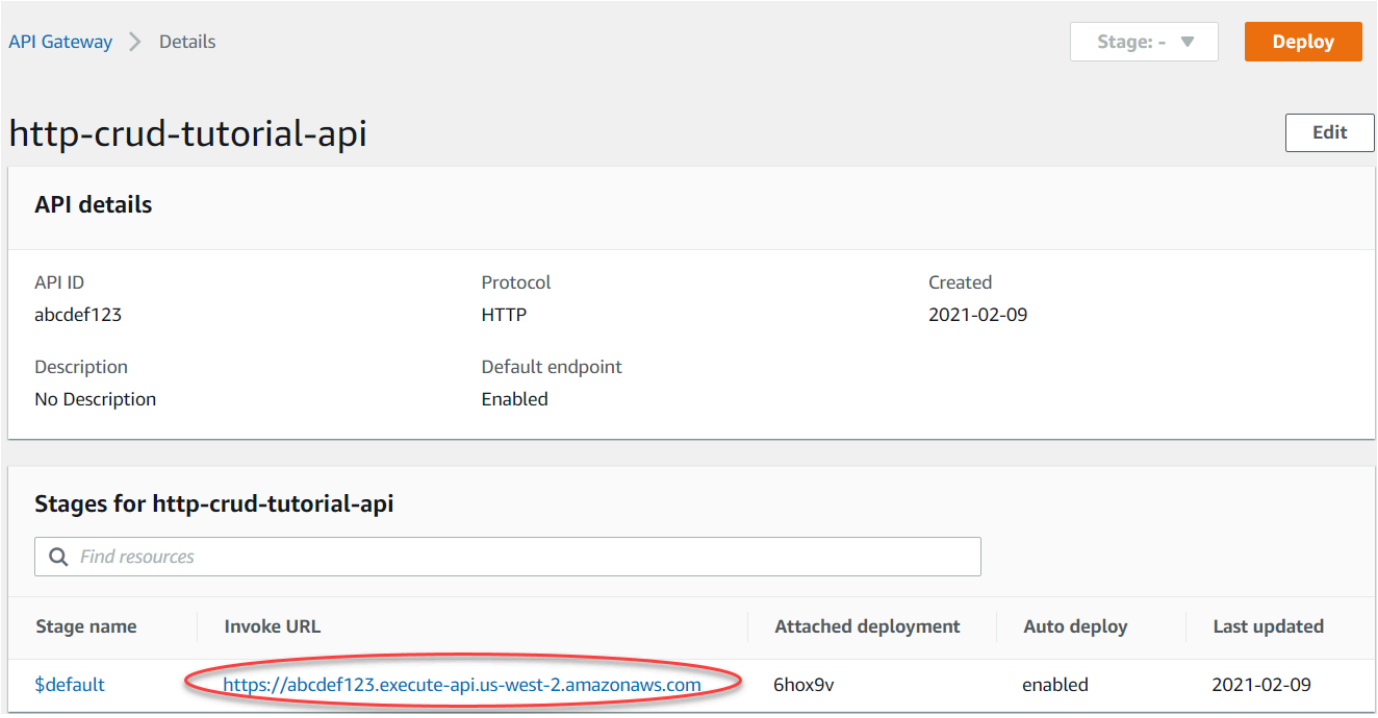
Jetzt, da Sie eine HTTP-API mit Routen und Integrationen haben, können Sie Ihre API testen.

Schritt 7: Ihre API testen

Um sicherzustellen, dass Ihre API funktioniert, verwenden Sie curl.

So wird die URL zum Aufrufen Ihrer API abgerufen

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Notieren Sie sich die Aufruf-URL Ihrer API. Sie wird auf der Seite Details unter Invoke URL (URL aufrufen) angezeigt.



4. Kopieren Sie die Aufruf-URL Ihrer API.

   Die vollständige URL sieht folgendermaßen aus: `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Erstellen oder Aktualisieren eines Elements

- Verwenden Sie den folgenden Befehl, um ein Element zu erstellen oder zu aktualisieren. Der Befehl enthält einen Anfragetext mit der ID, dem Preis und dem Namen des Elements.

```
curl -X "PUT" -H "Content-Type: application/json" -d "{\"id\": \"123\",
 \"price\": 12345, \"name\": \"myitem\"}" https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Abrufen aller Elemente

- Verwenden Sie den folgenden Befehl, um alle Elemente aufzulisten.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Abrufen eines Elements

- Verwenden Sie den folgenden Befehl, um ein Element anhand seiner ID zu erhalten.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

So löschen Sie einen Artikel

1. Verwenden Sie den folgenden Befehl, um ein Element zu löschen.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Rufen Sie alle Elemente ab, um zu überprüfen, ob das Element gelöscht wurde.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Schritt 8: Bereinigen

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

So löschen Sie eine DynamoDB-Tabelle

1. Öffnen Sie die DynamoDB-Konsole unter https://console.aws.amazon.com/dynamodb/.
2. Wählen Sie Ihre Tabelle aus.
3. Wählen Sie Delete Table (Tabelle löschen).
4. Bestätigen Sie Ihre Wahl und wählen Sie Delete (Löschen) aus.

So löschen Sie eine HTTP-API

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

So löschen Sie eine Lambda-Funktion

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

So löschen Sie die Protokollgruppe einer Lambda-Funktion

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

So löschen Sie die Ausführungsrolle einer Lambda-Funktion

1. Öffnen Sie in der AWS Identity and Access Management Konsole die Seite Rollen.
2. Wählen Sie die Rolle der Funktion aus, zum Beispie, `http-crud-tutorial-role`.
3. Wählen Sie Delete role (Rolle löschen) aus.
4. Wählen Sie Yes, delete (Ja, löschen) aus.

Nächste Schritte: Automatisieren Sie mit AWS SAM oder AWS CloudFormation

Sie können die Erstellung und Bereinigung von AWS Ressourcen automatisieren, indem Sie AWS CloudFormation oder AWS SAM verwenden. Ein Beispiel für eine AWS SAM -Vorlage für dieses Tutorial finden Sie unter `template.yaml`.

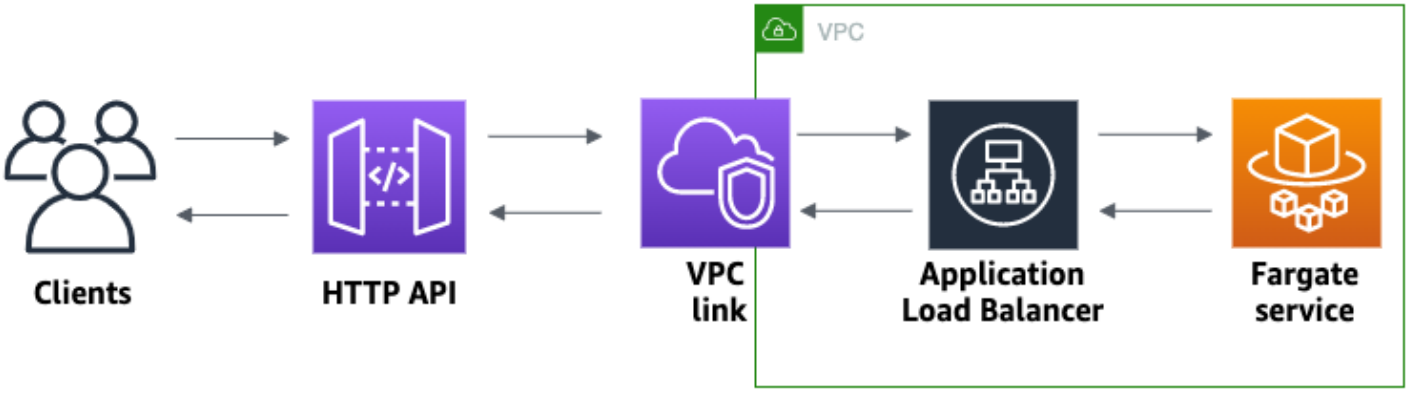
AWS CloudFormation Beispielvorlagen finden Sie unter AWS CloudFormation Beispielvorlagen.

Tutorial: Aufbau einer HTTP-API mit privater Integration in einen Amazon-ECS-Service

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

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

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



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

In diesem Tutorial wird AWS Management Console verwendet. Eine AWS CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter template.yaml.

Themen

Schritt 1: Erstellen Sie einen Amazon-ECS-Service

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

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

Um einen Stack zu erstellen AWS CloudFormation

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

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

Schritt 2: Erstellen eines VPC-Links

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

So erstellen Sie einen VPC-Link

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie im Hauptnavigationsbereich VPC-Links und dann Create aus.

   Möglicherweise müssen Sie das Menüsymbol auswählen, um den Hauptnavigationsbereich zu öffnen.
3. Wählen Sie unter VPC-Linkversion auswählen den VPC-Link für HTTP-APIs aus.
4. Geben Sie als Name die Zeichenfolge „**`private-integrations-tutorial`**“ ein.
5. Wählen Sie für VPC die in Schritt 1 erstellte VPC aus. Der Name sollte mit beginnen PrivateIntegrationsStack.
6. Wählen Sie für Subnetze die beiden privaten Subnetze in Ihrer VPC aus. Ihre Namen enden mit `PrivateSubnet`.
7. Wählen Sie Create aus.

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

Schritt 3: HTTP-API erstellen

Die HTTP-API bietet einen HTTP-Endpunkt für Ihre Amazon-ECS-Service-Funktion. In diesem Schritt erstellen Sie eine leere API. In den Schritten 4 und 5 konfigurieren Sie eine Route und eine Integration, um Ihre API und Ihren Amazon-ECS-Service miteinander zu verbinden.

So erstellen Sie eine HTTP-API

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) und dann für HTTP API die Option Build (Erstellen) aus.
3. Geben Sie unter API name (API-Name) **`http-private-integrations-tutorial`** ein.
4. Wählen Sie Next.
5. Wählen Sie für Configure routes (Routen konfigurieren) die Option Next (Weiter) aus, um die Routenerstellung zu überspringen. Sie werden Routen später erstellen.

6. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. API Gateway erstellt eine `$default` Phase mit aktivierten automatischen Bereitstellungen, was die beste Wahl für dieses Tutorial ist. Wählen Sie Next.
7. Wählen Sie Create aus.

Schritt 4: Erstellen einer Route

Routen sind eine Möglichkeit, eingehende API-Anforderungen an Backend-Ressourcen zu senden. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispie, `GET /items`. Für diese Beispiel-API erstellen wir eine Route:

Erstellen Sie eine Route.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wählen Sie Routes (Routen) aus.
4. Wählen Sie Create aus.
5. Wählen Sie für Methode **`ANY`** aus.
6. Für den Pfad geben Sie ei **`/{proxy+}`**. Das `{proxy+}` am Ende des Pfades ist eine „gierige“ Pfadvariable. API Gateway sendet alle Anfragen an Ihre API an diese Route.
7. Wählen Sie Create aus.

Schritt 5: Eine Integration erstellen

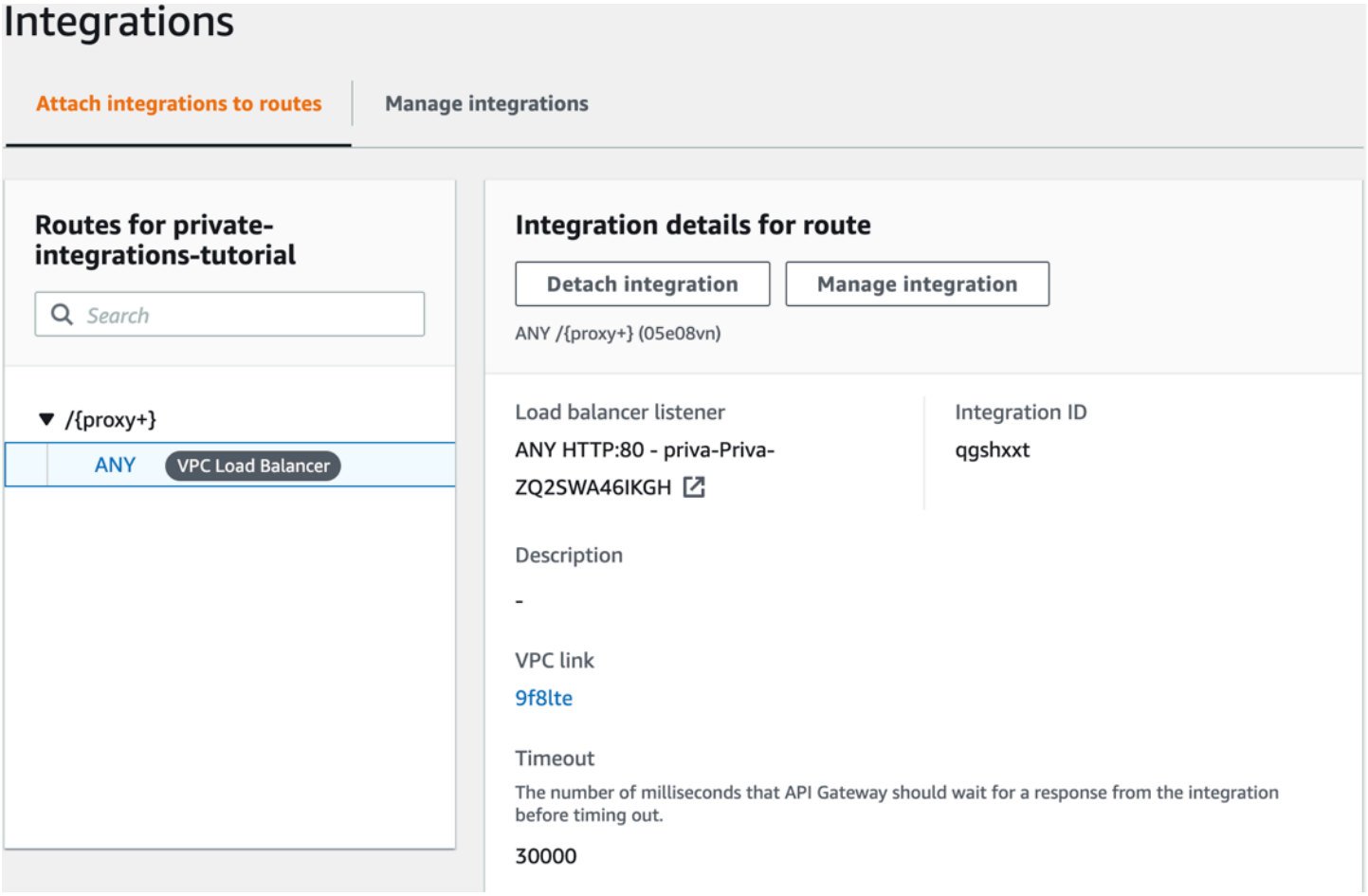
Sie erstellen eine Integration, um eine Route mit Backend-Ressourcen zu verbinden.

So wird eine Integration erstellt

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.
3. Wähle Sie Integrations (Integrationen) aus.
4. Wählen Sie Manage integrations (Integrationen verwalten) und dann Create (Erstellen) aus.
5. Wählen Sie für Diese Integration an eine Route anhängen die ANY/{proxy+}-Route aus, die Sie zuvor erstellt haben.

6. Für Integrationstyp wählen Sie Private Ressourceaus.
7. Für Integrationsdetails wählen Sie Manuell auswählen.
8. Für Ziel-Servicewählen Sie ALB/NLB.
9. Für Lastenverteilung wählen Sie die Lastenverteilung aus, die Sie mit der AWS CloudFormation -Vorlage in Schritt 1 erstellt haben. Der Name sollte mit HTTP-Priva beginnen.
10. Für den Listener wählen Sie **`HTTP 80`**.
11. Für VPC-Link wählen Sie den VPC-Link aus, den Sie in Schritt 2 erstellt haben. Ihr Name sollte laute `private-integrations-tutorial`.
12. Wählen Sie Create aus.

Um zu überprüfen, ob Ihre Route und Integration korrekt eingerichtet ist, wählen Sie Integrationen an Routen anhängen aus. Die Konsole zeigt, dass Sie eine `ANY /{proxy+}` Route mit einer Integration in einen VPC-Load-Balancer haben.



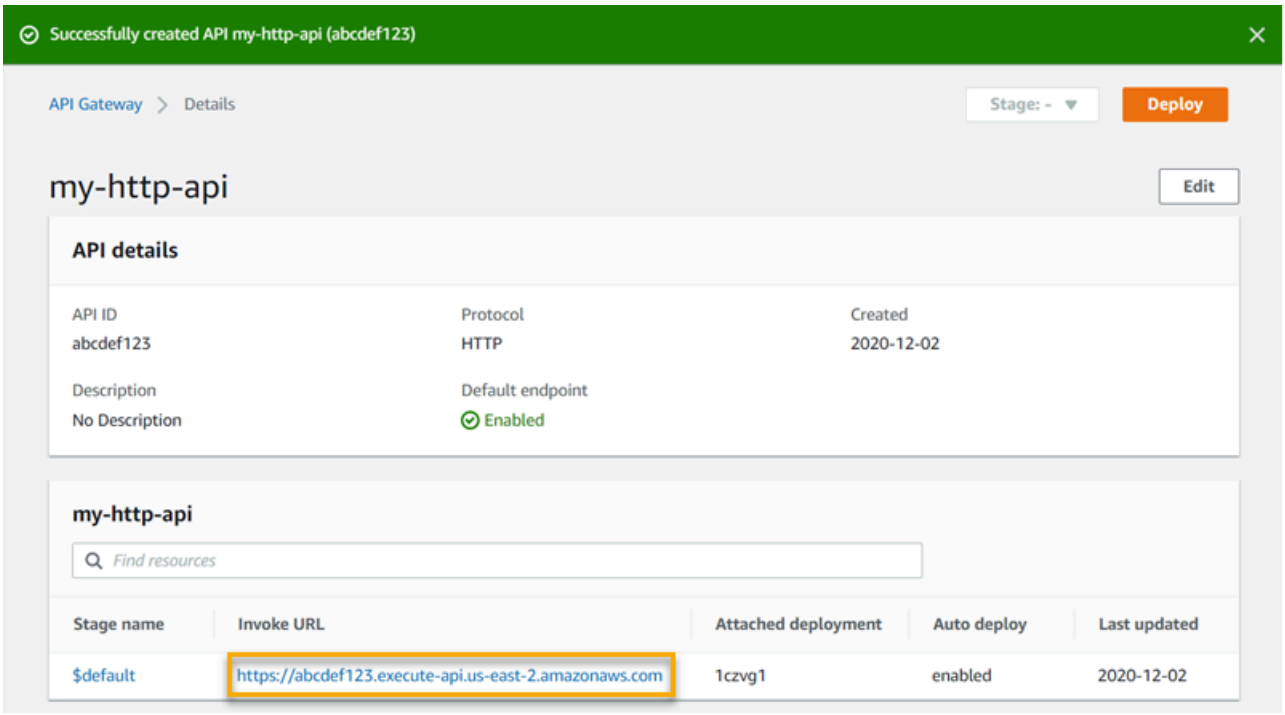
Jetzt sind Sie bereit, Ihre API zu testen.

Schritt 6: Ihre API testen

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

So testen Sie Ihre API

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



4. Rufen Sie in einem Webbrowser die Aufruf-URL Ihrer API auf.

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

   Ihr Browser sendet eine `GET`-Anforderung an die API.
5. Stellen Sie sicher, dass die Antwort Ihrer API eine Willkommensnachricht ist, die Ihnen mitteilt, dass Ihre App auf Amazon ECS ausgeführt wird.

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

Schritt 7: Bereinigen

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

So löschen Sie eine HTTP-API

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

So löschen Sie einen VPC-Link

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

Um einen Stack zu löschen AWS CloudFormation

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

Nächste Schritte: Automatisieren Sie mit AWS CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine vollständige AWS CloudFormation -Beispielvorlage finden Sie unter template.yaml.

Amazon API Gateway WebSocket API-Tutorials

Die folgenden Tutorials bieten praktische Übungen, die Ihnen helfen, mehr über API Gateway WebSocket Gateway-APIs zu erfahren.

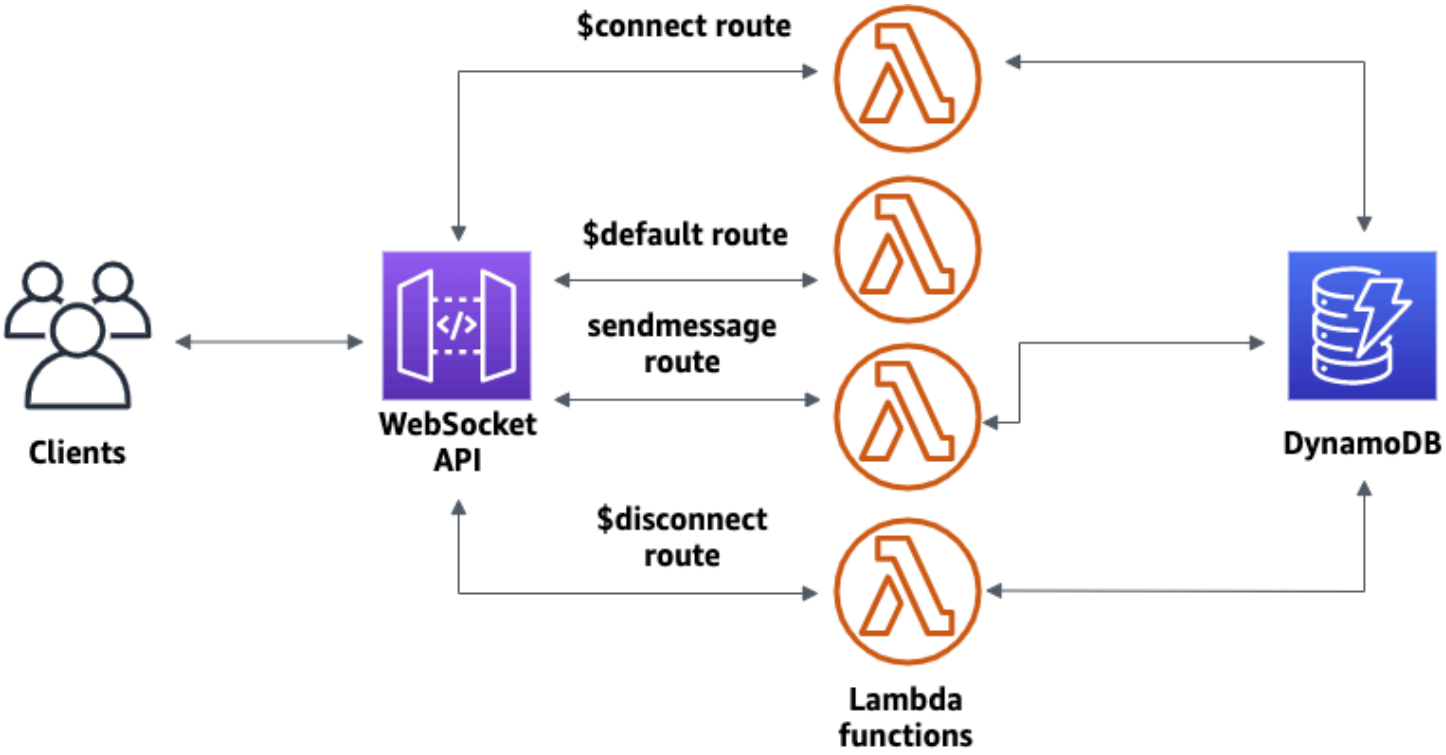
Themen

- Tutorial: Eine serverlose Chat-App mit einer WebSocket API, Lambda und DynamoDB erstellen
- Tutorial: Erstellen einer serverlosen Anwendung mit drei Integrationstypen

Tutorial: Eine serverlose Chat-App mit einer WebSocket API, Lambda und DynamoDB erstellen

In diesem Tutorial erstellen Sie eine serverlose Chat-Anwendung mit einer WebSocket API. Mit einer WebSocket API können Sie die bidirektionale Kommunikation zwischen Clients unterstützen. Kunden können Nachrichten erhalten, ohne Updates abfragen zu müssen.

Dieses Tutorial nimmt ungefähr 30 Minuten in Anspruch. Zunächst verwenden Sie eine AWS CloudFormation Vorlage, um Lambda-Funktionen zu erstellen, die API-Anfragen verarbeiten, sowie eine DynamoDB-Tabelle, in der Ihre Client-IDs gespeichert werden. Anschließend verwenden Sie die API Gateway Gateway-Konsole, um eine WebSocket API zu erstellen, die in Ihre Lambda-Funktionen integriert ist. Zuletzt testen Sie Ihre API, um zu prüfen, ob Nachrichten gesendet und empfangen werden.



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

Sie benötigen außerdem `wscat`, um eine Verbindung zu Ihrer API herzustellen. Weitere Informationen finden Sie unter the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.

Themen

- Schritt 1: Erstellen von Lambda-Funktionen und einer DynamoDB-Tabelle
- Schritt 2: Erstellen Sie eine API WebSocket
- Schritt 3: Testen Ihrer API
- Schritt 4: Bereinigen
- Nächste Schritte: Automatisieren Sie mit AWS CloudFormation

Schritt 1: Erstellen von Lambda-Funktionen und einer DynamoDB-Tabelle

Laden Sie die Vorlage zur App-Erstellung für AWS CloudFormation herunter und entpacken Sie sie. Sie verwenden diese Vorlage zur Erstellung einer Amazon-DynamoDB-Tabelle, in der die Client-IDs Ihrer App gespeichert werden. Jeder verbundene Client hat eine eindeutige ID, die wir als

Partitionsschlüssel der Tabelle verwenden werden. Diese Vorlage erstellt auch Lambda-Funktionen, die Ihre Client-Verbindungen in DynamoDB aktualisieren und das Senden von Nachrichten an verbundene Clients verarbeiten.

Um einen Stack zu erstellen AWS CloudFormation

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

AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres AWS CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen Sie eine API WebSocket

Sie erstellen eine WebSocket API, um Client-Verbindungen zu verarbeiten und Anfragen an die Lambda-Funktionen weiterzuleiten, die Sie in Schritt 1 erstellt haben.

Um eine API zu erstellen WebSocket

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) aus. Wählen Sie dann für WebSocket API die Option Build aus.
3. Geben Sie in API name (API-Name) **`websocket-chat-app-tutorial`** ein.

4. Geben Sie in Route selection expression (Ausdruck für die Routenauswahl) **`request.body.action`** ein. Der Ausdruck für die Routenauswahl legt die Route fest, die von API Gateway aufgerufen wird, wenn ein Client eine Nachricht sendet.
5. Wählen Sie Next (Weiter) aus.
6. Wählen Sie in Predefined routes (Vordefinierte Routen) die Optionen Add $connect ($connect hinzufügen), Add $disconnect ($disconnect hinzufügen) und Add $default ($default hinzufügen) aus. Die Routen $connect und $disconnect sind spezielle Routen, die API Gateway automatisch aufruft, wenn ein Client eine Verbindung zu einer API herstellt oder trennt. API Gateway ruft die Route `$default` auf, wenn keine anderen Routen mit einer Anforderung übereinstimmen.
7. Wählen Sie in Custom routes (Benutzerdefinierte Routen) Add custom route (Benutzerdefinierte Route hinzufügen) aus. Geben Sie in Route key (Routenschlüssel) **`sendmessage`** ein. Diese benutzerdefinierte Route verarbeitet Nachrichten, die an verbundene Clients gesendet werden.
8. Wählen Sie Weiter aus.
9. Wählen Sie in Attach integrations (Integrationen anfügen) für jede Route und jeden Integration type (Integrationstyp) „Lambda“ aus.

   Wählen Sie für Lambda die entsprechende Lambda-Funktion aus, mit der Sie AWS CloudFormation in Schritt 1 erstellt haben. Der Name jeder Funktion entspricht einer Route. Für die Route $connect wählen Sie beispielsweise die Funktion mit dem Namen **`websocket-chat-app-tutorial-ConnectHandler`** aus.
10. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. Standardmäßig erstellt API Gateway den Phasennamen `production` und stellt Ihre API automatisch für diese Phase bereit. Wählen Sie Weiter aus.
11. Wählen Sie Create and deploy (Erstellen und bereitstellen) aus.

Schritt 3: Testen Ihrer API

Als Nächstes testen Sie Ihre API, um sicherzustellen, dass sie korrekt funktioniert. Stellen Sie mit dem Befehl `wscat` eine Verbindung zur API her.

So rufen Sie die URL für den Aufruf Ihrer API ab

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API aus.

3. Wählen Sie Stages (Phasen) und anschließend production (Produktion) aus.
4. Notieren Sie sich die URL Ihrer API. WebSocket Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.

So stellen Sie eine Verbindung zu Ihrer API her

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zu Ihrer API her. Wenn Sie die Verbindung zu Ihrer API herstellen, ruft API Gateway die Route `$connect` auf. Bei Aufruf dieser Route wird eine Lambda-Funktion aufgerufen, mit der die Verbindungs-ID in DynamoDB gespeichert wird.

   ```
   wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
   ```

   ```
   Connected (press CTRL+C to quit)
   ```
2. Öffnen Sie ein neues Terminal. Führen Sie den Befehl wscat erneut mit den folgenden Parametern aus.

   ```
   wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
   ```

   ```
   Connected (press CTRL+C to quit)
   ```

   Anschließend erhalten Sie zwei verbundene Clients, die Nachrichten austauschen können.

So senden Sie eine Nachricht

- API Gateway legt basierend auf dem Ausdruck für die Routenauswahl Ihrer API fest, welche Route aufgerufen wird. Der Ausdruck für die Routenauswahl Ihrer API ist `$request.body.action`. Daher ruft API Gateway die Route `sendmessage` auf, wenn Sie die folgende Nachricht senden:

  ```
  {"action": "sendmessage", "message": "hello, everyone!"}
  ```

  Die Lambda-Funktion, die mit der aufgerufenen Route verknüpft ist, erfasst die Client-IDs aus DynamoDB. Anschließend ruft die Funktion die API-Gateway-Management-API auf und sendet die Nachricht an diese Clients. Alle verbundenen Clients erhalten die folgende Nachricht:

```
< hello, everyone!
```

So rufen Sie die Route „$default“ Ihrer API auf

- API Gateway ruft die Standardroute Ihrer API auf, wenn ein Client eine Nachricht sendet, die nicht mit den von Ihnen definierten Routen übereinstimmt. Die mit der Route `$default` verknüpfte Lambda-Funktion verwendet die API-Gateway-Management-API, um die Client-Daten zur Verbindung zu senden.

```
test
```

```
Use the sendmessage route to send a message. Your info:
 {"ConnectedAt":"2022-01-25T18:50:04.673Z","Identity":
{"SourceIp":"192.0.2.1","UserAgent":null},"LastActiveAt":"2022-01-25T18:50:07.642Z","connec
```

So trennen Sie die Verbindung zu Ihrer API

- Zur Trennung der Verbindung zu Ihrer API drücken Sie **`CTRL+C`**. Wenn ein Client die Verbindung zu Ihrer API trennt, ruft API Gateway die Route `$disconnect` Ihrer API auf. Die Lambda-Integration für die Route `$disconnect` Ihrer API entfernt die Verbindungs-ID aus DynamoDB.

Schritt 4: Bereinigen

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

Um eine WebSocket API zu löschen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie auf der Seite APIs Ihre `websocket-chat-app-tutorial`-API aus. Wählen Sie Actions (Aktionen) und Delete (Löschen) aus. Bestätigen Sie anschließend Ihre Auswahl.

Um einen AWS CloudFormation Stack zu löschen

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

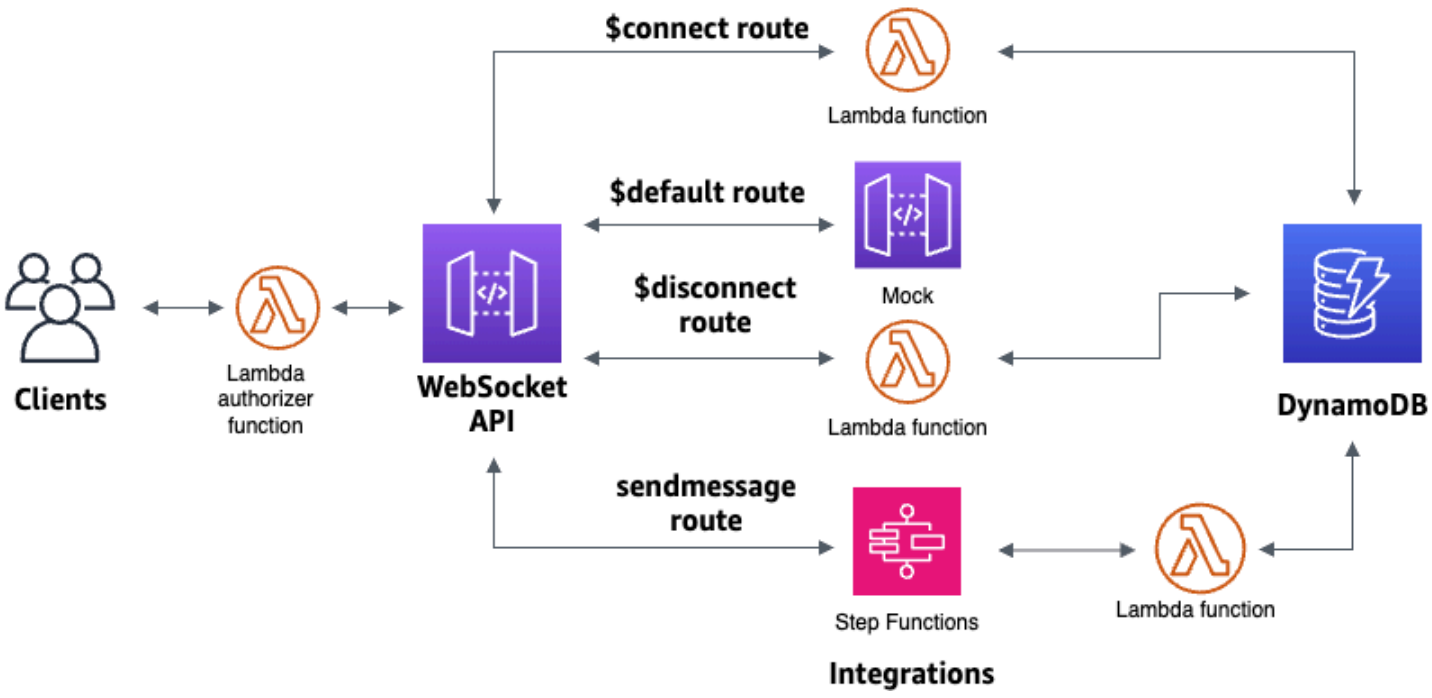
Nächste Schritte: Automatisieren Sie mit AWS CloudFormation

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Eine AWS CloudFormation Vorlage, die diese API und alle zugehörigen Ressourcen erstellt, finden Sie unter chat-app.yaml.

Tutorial: Erstellen einer serverlosen Anwendung mit drei Integrationstypen

In diesem Tutorial erstellen Sie eine serverlose Broadcast-Anwendung mit einer WebSocket API. Kunden können Nachrichten erhalten, ohne Updates abfragen zu müssen.

Dieses Tutorial zeigt, wie Nachrichten an verbundene Clients gesendet werden, und enthält ein Beispiel für einen Lambda-Authorizer, eine Scheinintegration und eine Nicht-Proxy-Integration in Step Functions.



Nachdem Sie Ihre Ressourcen mithilfe einer AWS CloudFormation Vorlage erstellt haben, verwenden Sie die API Gateway Gateway-Konsole, um eine WebSocket API zu erstellen, die sich in Ihre AWS Ressourcen integrieren lässt. Sie fügen Ihrer API einen Lambda-Authorizer hinzu und erstellen eine AWS Serviceintegration mit Step Functions, um eine State-Machine-Ausführung zu starten. Die Step Functions Functions-Zustandsmaschine ruft eine Lambda-Funktion auf, die eine Nachricht an alle verbundenen Clients sendet.

Nachdem Sie Ihre API erstellt haben, testen Sie Ihre Verbindung zu Ihrer API und stellen sicher, dass Nachrichten gesendet und empfangen werden. Die Fertigstellung dieses Tutorials dauert ungefähr 45 Minuten.

Themen

Voraussetzungen

Sie benötigen die folgenden Voraussetzungen:

- Ein AWS Konto und ein AWS Identity and Access Management Benutzer mit Konsolenzugriff. Weitere Informationen finden Sie unter Voraussetzungen.
- `wscat`um eine Verbindung zu Ihrer API herzustellen. Weitere Informationen finden Sie unter the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.

Wir empfehlen Ihnen, das WebSocket Chat-App-Tutorial abzuschließen, bevor Sie mit diesem Tutorial beginnen. Informationen zum Ausfüllen des WebSocket Chat-App-Tutorials finden Sie unterthe section called “WebSocket Chat-App”.

Schritt 1: Erstellen von Ressourcen

Laden Sie die Vorlage zur App-Erstellung für AWS CloudFormation herunter und entpacken Sie sie. Sie werden diese Vorlage verwenden, um Folgendes zu erstellen:

- Lambda-Funktionen, die API-Anfragen bearbeiten und den Zugriff auf Ihre API autorisieren.
- Eine DynamoDB-Tabelle zum Speichern von Client-IDs und der vom Lambda-Autorisierer zurückgegebenen Hauptbenutzeridentifikation.
- Eine Step Functions Functions-Zustandsmaschine zum Senden von Nachrichten an verbundene Clients.

Um einen AWS CloudFormation Stapel zu erstellen

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

AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wählen Sie den Tab Ausgaben, um Ihre erstellten Ressourcen und deren ARNs zu sehen. Wenn der Status Ihres AWS CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

Schritt 2: Erstellen Sie eine API WebSocket

Sie erstellen eine WebSocket API, um Client-Verbindungen zu verarbeiten und Anfragen an die Ressourcen weiterzuleiten, die Sie in Schritt 1 erstellt haben.

Um eine WebSocket API zu erstellen

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) aus. Wählen Sie dann für WebSocket API die Option Build aus.
3. Geben Sie in API name (API-Name) **`websocket-step-functions-tutorial`** ein.
4. Geben Sie in Route selection expression (Ausdruck für die Routenauswahl) **`request.body.action`** ein.

   Der Ausdruck für die Routenauswahl legt die Route fest, die von API Gateway aufgerufen wird, wenn ein Client eine Nachricht sendet.
5. Wählen Sie Next (Weiter) aus.
6. Wählen Sie für vordefinierte Routen Add $connect, Add $disconnect, Add $default.

   Die Routen $connect und $disconnect sind spezielle Routen, die API Gateway automatisch aufruft, wenn ein Client eine Verbindung zu einer API herstellt oder trennt. API Gateway ruft die $default Route auf, wenn keine anderen Routen mit einer Anfrage übereinstimmen. Nachdem Sie Ihre API erstellt haben, erstellen Sie eine benutzerdefinierte Route, um eine Verbindung zu Step Functions herzustellen.
7. Wählen Sie Weiter aus.
8. Gehen Sie für die Integration für $connect wie folgt vor:

   a. Wählen Sie als Integrationstyp Lambda aus.

   b. Wählen Sie für die Lambda-Funktion die entsprechende $connect Lambda-Funktion aus, mit der Sie AWS CloudFormation in Schritt 1 erstellt haben. Der Name der Lambda-Funktion sollte mit **`websocket-step`** beginnen.
9. Gehen Sie für die Integration für $disconnect wie folgt vor:

   a. Wählen Sie als Integrationstyp Lambda aus.

   b. Wählen Sie für die Lambda-Funktion die entsprechende $disconnect Lambda-Funktion aus, mit der Sie AWS CloudFormation in Schritt 1 erstellt haben. Der Name der Lambda-Funktion sollte mit **`websocket-step`** beginnen.
10. Wählen Sie für Integration for $default die Option mock aus.

Bei einer Scheinintegration verwaltet API Gateway die Routenantwort ohne ein Integrations-Backend.

11. Wählen Sie Weiter aus.
12. Überprüfen Sie die Phase, die API Gateway für Sie erstellt. Standardmäßig erstellt API Gateway eine Phase mit dem Namen production und stellt Ihre API automatisch in dieser Phase bereit. Wählen Sie Weiter aus.
13. Wählen Sie Create and deploy (Erstellen und bereitstellen) aus.

Schritt 3: Erstellen Sie einen Lambda-Autorisierer

Um den Zugriff auf Ihre WebSocket API zu kontrollieren, erstellen Sie einen Lambda-Autorisierer. Die AWS CloudFormation Vorlage hat die Lambda-Autorisierungsfunktion für Sie erstellt. Sie können die Lambda-Funktion in der Lambda-Konsole sehen. Der Name sollte mit beginnen. **`websocket-step-functions-tutorial-AuthorizerHandler`** Diese Lambda-Funktion verweigert alle WebSocket API-Aufrufe, es sei denn, der `Authorization` Header ist es. `Allow` Die Lambda-Funktion übergibt die `$context.authorizer.principalId` Variable auch an Ihre API, die später in der DynamoDB-Tabelle verwendet wird, um API-Aufrufer zu identifizieren.

In diesem Schritt konfigurieren Sie die $connect-Route für die Verwendung des Lambda-Autorisierers.

Um einen Lambda-Autorisierer zu erstellen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Wählen Sie Create an Authorizer aus.
4. Geben Sie als Namen des Autorisierers ein. **`LambdaAuthorizer`**
5. Geben Sie für Authorizer ARN den Namen des Autorisierers ein, der mit der AWS CloudFormation Vorlage erstellt wurde. Der Name sollte mit beginnen. **`websocket-step-functions-tutorial-AuthorizerHandler`**

> **Note**
>
> Wir empfehlen, diesen Beispiel-Authorizer nicht für Ihre Produktions-APIs zu verwenden.

6. Wählen Sie als Identitätsquellentyp die Option Header aus. Geben Sie für Key (Schlüssel) **`Authorization`** ein.
7. Wählen Sie Autorisierer erstellen.

Nachdem Sie Ihren Authorizer erstellt haben, hängen Sie ihn an die $connect-Route Ihrer API an.

Um einen Authorizer an die $connect-Route anzuhängen

1. Wählen Sie im Hauptnavigationsbereich Routes aus.
2. Wählen Sie die $connect-Route.
3. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Autorisierung das Drop-down-Menü und wählen Sie dann Ihren Anforderungsautorisierer aus.
5. Wählen Sie Änderungen speichern aus.

Schritt 4: Erstellen Sie eine simulierte bidirektionale Integration

Als Nächstes erstellen Sie die bidirektionale Scheinintegration für die $default-Route. Mit einer Scheinintegration können Sie eine Antwort an den Client senden, ohne ein Backend zu verwenden. Wenn Sie eine Integration für die $default Route erstellen, können Sie Kunden zeigen, wie sie mit Ihrer API interagieren können.

Sie konfigurieren die $default Route, um Clients darüber zu informieren, dass sie die SendMessage-Route verwenden sollen.

Um eine Scheinintegration zu erstellen

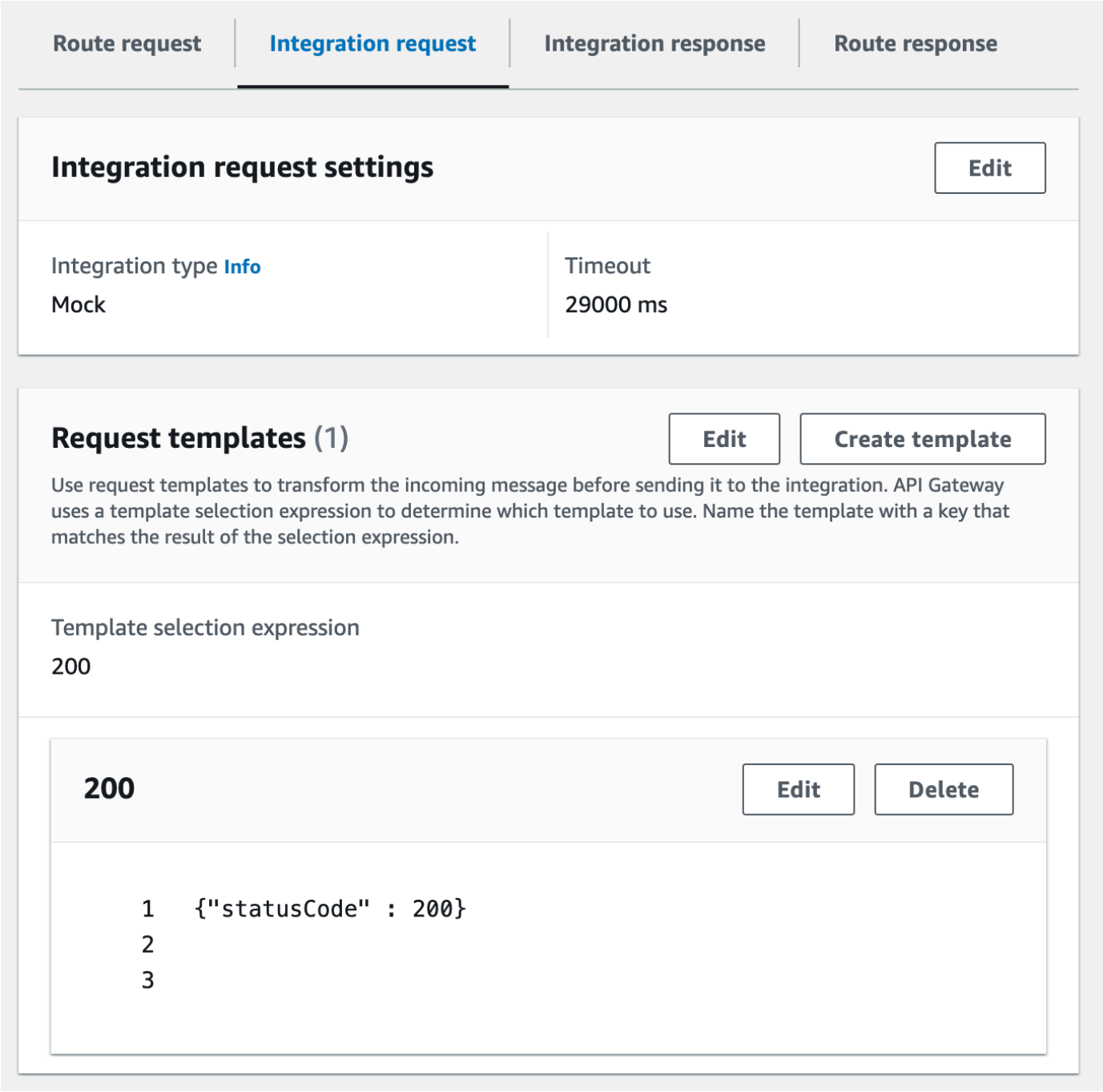
1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie die $default Route und dann den Tab Integrationsanfrage.
3. Wählen Sie für Anforderungsvorlagen die Option Bearbeiten aus.
4. Geben Sie als Ausdruck für die Vorlagenauswahl den **`200`** Wert ein, und wählen Sie dann Bearbeiten aus.
5. Wählen Sie auf der Registerkarte Integrationsanfrage für Anforderungsvorlagen die Option Vorlage erstellen aus.

6. Geben Sie als Vorlagenschlüssel ein**200**.
7. Geben Sie für Vorlage generieren die folgende Zuordnungsvorlage ein:

```
{"statusCode": 200}
```

Wählen Sie Create template (Vorlage erstellen) aus.

Das Ergebnis sollte wie folgt aussehen:

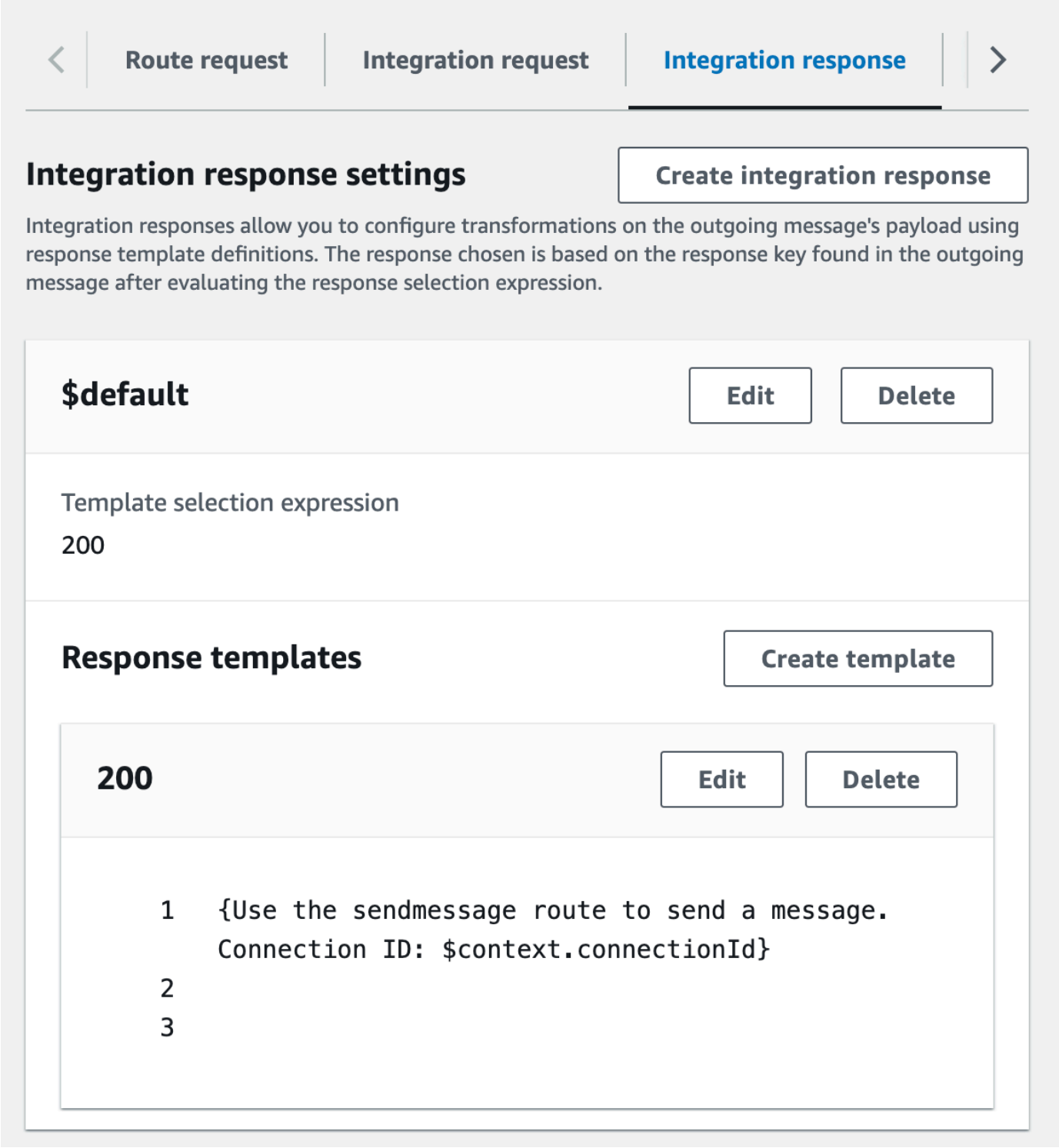


8. Wählen Sie im Bereich $default Route die Option Bidirektionale Kommunikation aktivieren aus.
9. Wählen Sie die Registerkarte Integrationsantwort und dann Integrationsantwort erstellen aus.
10. Geben Sie als Antwortschlüssel ein**`$default`**.
11. Geben Sie als Ausdruck für die Vorlagenauswahl den Wert ein**`200`**.
12. Wählen Sie Create response (Antwort erstellen) aus.
13. Wählen Sie unter Antwortvorlagen die Option Vorlage erstellen aus.
14. Geben Sie als Vorlagenschlüssel ein**`200`**.
15. Geben Sie für Antwortvorlage die folgende Zuordnungsvorlage ein:

```
{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}
```

16. Wählen Sie Create template (Vorlage erstellen) aus.

    Das Ergebnis sollte wie folgt aussehen:



Schritt 5: Erstellen Sie eine Nicht-Proxy-Integration mit Step Functions

Als Nächstes erstellen Sie eine SendMessage-Route. Clients können die SendMessage-Route aufrufen, um eine Nachricht an alle verbundenen Clients zu senden. Die SendMessage-Route

hat eine Nicht-Proxy-Serviceintegration mit. AWS AWS Step Functions Die Integration ruft den StartExecutionBefehl für den Step Functions Functions-Zustandsmaschine auf, den die AWS CloudFormation Vorlage für Sie erstellt hat.

Um eine Nicht-Proxy-Integration zu erstellen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie in Route key (Routenschlüssel) **`sendmessage`** ein.
4. Wählen Sie als Integrationstyp die Option AWS Service aus.
5. Geben Sie für AWS Region die Region ein, in der Sie Ihre AWS CloudFormation Vorlage bereitgestellt haben.
6. Wählen Sie für AWS Service Step Functions.
7. Wählen Sie in HTTP method POST.
8. Für Aktionsname geben Sie **`StartExecution`** ein.
9. Geben Sie für Ausführungsrolle die Ausführungsrolle ein, die durch die AWS CloudFormation Vorlage erstellt wurde. Der Name sollte sein WebsocketTutorialApiRole.
10. Wählen Sie Create route (Route erstellen) aus.

Als Nächstes erstellen Sie eine Mapping-Vorlage, um Anforderungsparameter an die Step Functions Functions-Zustandsmaschine zu senden.

Erstellen einer Zuweisungsvorlage

1. Wählen Sie die SendMessage-Route und dann die Registerkarte Integrationsanfrage.
2. Wählen Sie im Abschnitt Anforderungsvorlagen die Option Bearbeiten aus.
3. Geben Sie als Ausdruck für die Vorlagenauswahl den Wert ein**`\$default`**.
4. Wählen Sie Bearbeiten aus.
5. Wählen Sie im Abschnitt Vorlagen anfordern die Option Vorlage erstellen aus.
6. Geben Sie als Vorlagenschlüssel ein**`\$default`**.
7. Geben Sie für Vorlage generieren die folgende Zuordnungsvorlage ein:

```
#set($domain = "$context.domainName")
#set($stage = "$context.stage")
```

```
#set($body = $input.json('$'))
#set($getMessage = $util.parseJson($body))
#set($mymessage = $getMessage.message)
{
"input": "{\"domain\": \"$domain\", \"stage\": \"$stage\", \"message\":
 \"$mymessage\"}",
"stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:WebSocket-
Tutorial-StateMachine"
}
```

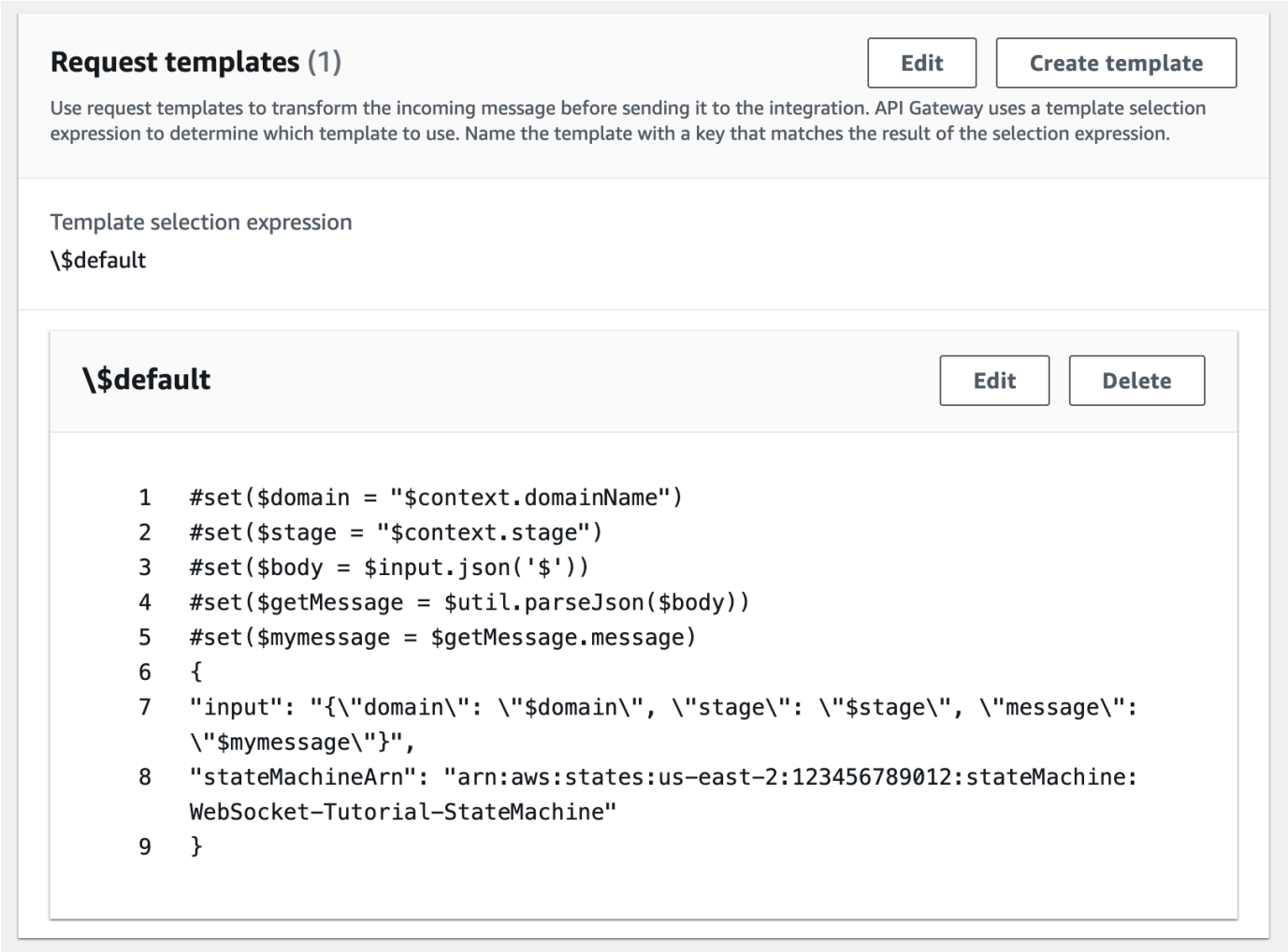
Ersetzen Sie das *stateMachineArn*durch den ARN der Zustandsmaschine, die von erstellt wurde AWS CloudFormation.

Die Zuordnungsvorlage macht Folgendes:

- Erzeugt die Variable `$domain` mithilfe der Kontextvariablen`domainName`.
- Erzeugt die Variable `$stage` mithilfe der Kontextvariablen`stage`.

  Die `$stage` Variablen `$domain` und sind erforderlich, um eine Callback-URL zu erstellen.
- Nimmt die eingehende `sendmessage` JSON-Nachricht auf und extrahiert die `message` Eigenschaft.
- Erzeugt die Eingabe für die Zustandsmaschine. Die Eingabe ist die Domäne und die Phase der WebSocket API sowie die Nachricht von der `sendmessage` Route.

8. Wählen Sie Create template (Vorlage erstellen) aus.



Sie können eine Nicht-Proxy-Integration auf den Routen $connect oder $disconnect erstellen, um direkt eine Verbindungs-ID zur DynamoDB-Tabelle hinzuzufügen oder zu entfernen, ohne eine Lambda-Funktion aufzurufen.

Schritt 6: Ihre API testen

Als Nächstes stellen Sie Ihre API bereit und testen sie, um sicherzustellen, dass sie ordnungsgemäß funktioniert. Sie verwenden den `wscat` Befehl, um eine Verbindung zur API herzustellen, und dann verwenden Sie einen Slash-Befehl, um einen Ping-Frame zu senden, um die Verbindung zur WebSocket API zu überprüfen.

Stellen Sie Ihre API bereit

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Wählen Sie im Hauptnavigationsbereich Routes aus.
3. Klicken Sie auf Deploy API.
4. Wählen Sie für Stage die Option Production aus.
5. (Optional) Geben Sie unter Beschreibung der Bereitstellung eine Beschreibung ein.
6. Wählen Sie Bereitstellen.

Nachdem Sie Ihre API bereitgestellt haben, können Sie sie aufrufen. Verwenden Sie die Aufruf-URL, um Ihre API aufzurufen.

Um die Aufruf-URL für Ihre API abzurufen

1. Wählen Sie Ihre API aus.
2. Wählen Sie Stages (Phasen) und anschließend production (Produktion) aus.
3. Notieren Sie sich die WebSocket URL Ihrer API. Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.

Jetzt, da Sie Ihre Aufruf-URL haben, können Sie die Verbindung zu Ihrer WebSocket API testen.

Um die Verbindung zu Ihrer API zu testen

1. Stellen Sie mit dem folgenden Befehl eine Verbindung zu Ihrer API her. Zuerst testen Sie die Verbindung, indem Sie den `/ping` Pfad aufrufen.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Geben Sie den folgenden Befehl ein, um den Kontrollrahmen zu pingen. Sie können einen Kontrollrahmen für Keepalive-Zwecke von der Clientseite aus verwenden.

```
/ping
```

   Das Ergebnis sollte wie folgt aussehen:

```
< Received pong (data: "")
```

Nachdem Sie die Verbindung getestet haben, können Sie testen, ob Ihre API ordnungsgemäß funktioniert. In diesem Schritt öffnen Sie ein neues Terminalfenster, damit die WebSocket API eine Nachricht an alle verbundenen Clients senden kann.

So testen Sie Ihre API

1. Öffnen Sie ein neues Terminal. Führen Sie den Befehl `wscat` erneut mit den folgenden Parametern aus.

   ```
   wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
    "Authorization: Allow"
   ```

   ```
   Connected (press CTRL+C to quit)
   ```

2. API Gateway bestimmt anhand des Ausdrucks zur Routenanforderungsauswahl Ihrer API, welche Route aufgerufen werden soll. Der Route-Select-Ausdruck Ihrer API lautet`$request.body.action`. Daher ruft API Gateway die Route `sendmessage` auf, wenn Sie die folgende Nachricht senden:

   ```
   {"action": "sendmessage", "message": "hello, from Step Functions!"}
   ```

   Die der Route zugeordnete Step Functions Functions-Zustandsmaschine ruft eine Lambda-Funktion mit der Nachricht und der Callback-URL auf. Die Lambda-Funktion ruft die API Gateway Management API auf und sendet die Nachricht an alle verbundenen Clients. Alle Clients erhalten die folgende Nachricht:

   ```
   < hello, from Step Functions!
   ```

Nachdem Sie Ihre WebSocket API getestet haben, können Sie die Verbindung zu Ihrer API trennen.

So trennen Sie die Verbindung zu Ihrer API

- Zur Trennung der Verbindung zu Ihrer API drücken Sie `CTRL+C`.

  Wenn ein Client die Verbindung zu Ihrer API trennt, ruft API Gateway die $disconnect-Route Ihrer API auf. Die Lambda-Integration für die $disconnect-Route Ihrer API entfernt die Verbindungs-ID aus DynamoDB.

Schritt 7: Bereinigen

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

Um eine WebSocket API zu löschen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie auf der API-Seite Ihre Websocket-API aus.
3. Wählen Sie Actions (Aktionen) und Delete (Löschen) aus. Bestätigen Sie anschließend Ihre Auswahl.

Um einen Stapel zu löschen AWS CloudFormation

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

Nächste Schritte

Sie können die Erstellung und Bereinigung aller an diesem Tutorial beteiligten AWS Ressourcen automatisieren. Ein Beispiel für eine AWS CloudFormation Vorlage, die diese Aktionen für dieses Tutorial automatisiert, finden Sie unter ws-sfn.zip.

Mit REST-APIs arbeiten

Eine REST-API in API Gateway ist eine Sammlung von Ressourcen und Methoden, die in Backend-HTTP-Endpunkte, Lambda-Funktionen oder andere Dienste integriert sind. AWS Sie können API Gateway-Funktionen verwenden, um Ihnen bei allen Aspekten des API-Lebenszyklus zu helfen, von der Erstellung bis zur Überwachung Ihrer Produktions-APIs.

API Gateway-REST-APIs verwenden ein Anfrage-/Antwortmodell, bei dem ein Client eine Anfrage an einen Service sendet und der Service synchron zurückantwortet. Diese Art von Modell eignet sich für viele verschiedene Arten von Anwendungen, die von synchroner Kommunikation abhängen.

Themen

REST-API in API Gateway entwickeln

In Amazon API Gateway erstellen Sie eine REST-API als eine Sammlung programmierbarer Entitäten, die als API Gateway-Ressourcen bekannt ist. Sie verwenden beispielsweise eine RestApiRessource, um eine API darzustellen, die eine Sammlung von Ressourcenentitäten enthalten kann.

Jede `Resource` Entität kann über eine oder mehrere Methodenressourcen verfügen. A `Method` ist eine eingehende Anfrage, die vom Client eingereicht wird und in den Anforderungsparametern und dem Text ausgedrückt wird. Es definiert die Anwendungsprogrammierschnittstelle, über die der Client auf die exponierten Dateien zugreifen kann`Resource`. Um den `Method` in einen Backend-Endpunkt, der auch als Integrationsendpunkt bezeichnet wird, zu integrieren, erstellen Sie eine Integrationsressource. Dadurch wird die eingehende Anfrage an einen angegebenen Integrationsendpunkt-URI weitergeleitet. Bei Bedarf können Sie die Anforderungsparameter oder den Anforderungstext transformieren, um die Backend-Anforderungen zu erfüllen.

Für Antworten können Sie eine MethodResponseRessource erstellen, die eine vom Client empfangene Anforderungsantwort darstellt, und Sie können eine IntegrationResponseRessource erstellen, die die vom Backend zurückgegebene Anforderungsantwort darstellt. Sie können die Integrationsantwort konfigurieren, um die Backend-Antwortdaten zu transformieren, bevor die Daten an den Client zurückgegeben werden, oder um die Backend-Antwort unverändert an den Client zu übermitteln.

Um Ihren Kunden das Verständnis Ihrer API zu erleichtern, können Sie auch eine Dokumentation für die API, im Rahmen der API-Erstellung oder nachdem die API erstellt wurde, bereitstellen. Um dies zu aktivieren, fügen Sie eine DocumentationPartRessource für eine unterstützte API-Entität hinzu.

Um zu steuern, wie Clients eine API aufrufen, verwenden Sie IAM-Berechtigungen, einen Lambda-Genehmiger oder einen Amazon Cognito-Benutzerpool. Um die Nutzung Ihrer API zu messen, richten Sie Nutzungspläne ein, um die API-Anforderungen zu drosseln. Sie können diese aktivieren, wenn Sie Ihre API erstellen oder aktualisieren.

Eine Einführung in die Erstellung einer API finden Sie unterthe section called “Tutorial: Hello World-API mit Lambda-Proxy-Integration erstellen”. Weitere Informationen zu den Funktionen von API Gateway, die Sie möglicherweise bei der Entwicklung einer REST-API verwenden, finden Sie in den folgenden Themen. Diese Themen enthalten konzeptionelle Informationen und Verfahren, die Sie mit der API Gateway Gateway-Konsole, der API Gateway Gateway-REST-API AWS CLI, dem oder einem der AWS SDKs ausführen können.

Themen

- API-Gateway-API-Endpunkttypen
- Methoden für REST-APIs in API Gateway
- Zugriff auf eine REST-API in API Gateway steuern und verwalten
- REST-API-Integrationen einrichten
- Verwenden der Anforderungsvalidierung in API Gateway
- Datentransformationen für REST-APIs einrichten
- Gateway-Antworten in API Gateway
- Aktivieren von CORS für eine REST-API-Ressource
- Mit binären Medientypen für REST-APIs arbeiten
- REST-API im Amazon API Gateway aufrufen
- Konfigurieren einer REST-API mit OpenAPI

API-Gateway-API-Endpunkttypen

Ein API-Endpunkt-Typ bezieht sich auf den Hostnamen der API. Der API-Endpunkttyp kann Edge-optimiert, regional oder privat sein, je nachdem, woher der Großteil Ihres API-Datenverkehrs stammt.

Edge-optimierte API-Endpunkte

Ein Edge-optimierter API-Endpunkt leitet Anfragen in der Regel an den nächstgelegenen CloudFront Point of Presence (POP) weiter, was in Fällen hilfreich sein kann, in denen Ihre Kunden geografisch verteilt sind. Dies ist der Standard-Endpunkttyp für API Gateway-REST-APIs.

Edge-optimierte APIs profitieren von den Namen der HTTP-Header (z. B. `Cookie`).

CloudFront sortiert HTTP-Cookies in natürlicher Reihenfolge nach dem Namen des Cookies, bevor die Anfrage an Ihren Absender weitergeleitet wird. Weitere Informationen zur Art und Weise, wie Cookies CloudFront verarbeitet werden, finden Sie unter Zwischenspeichern von Inhalten auf der Grundlage von Cookies.

Jeder von Ihnen für eine Edge-optimierte API verwendeter benutzerdefinierter Domänenname gilt für alle Regionen.

Regionale API-Endpunkte

Ein regionaler API-Endpunkt ist für Clients in derselben Region vorgesehen. Wenn ein Client, der auf einer EC2-Instance ausgeführt wird, eine API in derselben Region aufruft oder wenn eine API eine kleine Anzahl von Clients mit hohen Anforderungen bedienen soll, reduziert eine regionale API den Verbindungsaufwand.

Bei einer regionalen API ist jeder benutzerdefinierte Domainname, den Sie verwenden, spezifisch für die Region, in der die API bereitgestellt wird. Wenn Sie eine regionale API in mehreren Regionen bereitstellen, kann sie in allen Regionen den gleichen Domänennamen haben. Sie können benutzerdefinierte Domänen zusammen mit Amazon Route 53 verwenden, um Aufgaben wie latenzbasiertes Routing auszuführen. Weitere Informationen erhalten Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens” und the section called “Erstellen eines Edge-optimierten benutzerdefinierten Domänennamens”.

Regionale API-Endpunkte übergeben alle Headernamen unverändert.

> **Note**
>
> In Fällen, in denen API-Clients geografisch verstreut sind, kann es dennoch sinnvoll sein, einen regionalen API-Endpunkt zusammen mit Ihrer eigenen CloudFront Amazon-Distribution zu verwenden, um sicherzustellen, dass API Gateway die API nicht mit CloudFront servicegesteuerten Distributionen verknüpft. Weitere Informationen zu diesem Anwendungsfall finden Sie unter Wie richte ich API Gateway mit meiner eigenen CloudFront Distribution ein? .

Private API-Endpunkte

Ein privater API-Endpunkt ist ein API-Endpunkt, auf den nur von Ihrer Amazon Virtual Private Cloud (VPC) aus zugegriffen werden kann, und zwar über einen Interface-VPC-Endpunkt, bei dem es sich um eine Endpunkt-Netzwerkschnittstelle (ENI) handelt, die Sie in Ihrer VPC erstellen. Weitere Informationen finden Sie unter the section called “Private REST-APIs”.

Private API-Endpunkte übergeben alle Headernamen unverändert.

Öffentlichen oder privaten API-Endpunkttyp in API Gateway ändern

Das Ändern eines API-Endpunkttyps erfordert, dass Sie die Konfiguration der API aktualisieren. Sie können einen vorhandenen API-Typ mit der API Gateway-Konsole AWS CLI, dem oder einem AWS SDK für API Gateway ändern. Der Endpunkttyp kann erst wieder geändert werden, wenn die aktuelle Änderung abgeschlossen ist, aber Ihre API ist verfügbar.

Die folgenden Änderungen am Endpunkttyp werden nicht unterstützt:

- Von Edge-Optimized zu Regional oder Private
- Von regional bis Edge-optimiert oder privat
- Von privat zu regional

Sie können eine private API nicht in eine Edge-optimierte API ändern.

Wenn Sie eine öffentliche API von Edge-Optimized auf Regional oder umgekehrt ändern, beachten Sie, dass sich eine Edge-optimierte API möglicherweise anders verhält als eine regionale API. Bei einer Edge-optimierten API wird beispielsweise der `Content-MD5`-Header entfernt. Ein an das Backend übergebener MD5-Hash-Wert kann in einem Anforderungszeichenfolgenparameter oder einer Texteigenschaft dargestellt werden. Die regionale API leitet diesen Header jedoch weiter,

obwohl sie den Header-Namen möglicherweise einem anderen Namen zuordnet. Wenn Sie die Unterschiede kennen, können Sie leichter entscheiden, wie Sie eine Edge-optimierte API auf eine regionale oder von einer regionalen API auf eine Edge-optimierte API aktualisieren möchten.

Themen

API-Endpunkttyp über die API Gateway-Konsole ändern

Um den API-Endpunkttyp Ihrer API zu ändern, führen Sie einen der folgenden Schritte aus:

So wandeln Sie einen öffentlichen Endpunkt von regional zu Edge-optimiert oder umgekehrt um

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie API-Einstellungen aus.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Wählen Sie als API-Endpunkttyp entweder Edge-optimiert oder Regional aus.
6. Wählen Sie Änderungen speichern aus.
7. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

So konvertieren Sie einen privaten Endpunkt in einen regionalen Endpunkt

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Bearbeiten Sie die Ressourcenrichtlinie für Ihre API und entfernen Sie alle Hinweise auf VPCs oder VPC-Endpunkte, sodass API-Aufrufe von außerhalb Ihrer VPC als auch innerhalb Ihrer VPC erfolgreich ausgeführt werden können.
4. Wählen Sie API-Einstellungen aus.
5. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
6. Wählen Sie für API-Endpunkttyp die Option Regional aus.
7. Wählen Sie Änderungen speichern aus.

8. Entfernen Sie die Ressourcenrichtlinie aus Ihrer API.
9. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

Um einen regionalen Endpunkt in einen privaten Endpunkt zu konvertieren

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Erstellen Sie eine Ressourcenrichtlinie, die Zugriff auf Ihre VPC oder Ihren VPC-Endpunkt gewährt. Weitere Informationen finden Sie unter ???.
4. Wählen Sie API-Einstellungen aus.
5. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
6. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
7. (Optional) Wählen Sie für VPC-Endpunkt-IDs die VPC-Endpunkt-IDs aus, die Sie Ihrer privaten API zuordnen möchten.
8. Wählen Sie Änderungen speichern aus.
9. Stellen Sie die API erneut bereit, sodass die Änderungen wirksam werden.

Verwenden Sie die AWS CLI , um den Typ eines API-Endpunkts zu ändern

Rufen Sie wie folgt auf AWS CLI , um eine Edge-optimierte API mit der API-ID zu aktualisieren: *{api-id}* update-rest-api

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
    --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

Die erfolgreiche Antwort verfügt über einen Statuscode von `200 OK` und eine Nutzlast, die etwa wie folgt aussieht:

```
{

    "createdDate": "2017-10-16T04:09:31Z",
    "description": "Your first API with Amazon API Gateway. This is a sample API that
 integrates via HTTP with our demo Pet Store endpoints",
    "endpointConfiguration": {
```

```
        "types": "REGIONAL"
    },
    "id": "0gsnjtjck8",
    "name": "PetStore imported as edge-optimized"
}
```

Wandeln Sie umgekehrt eine regionale API wie folgt in eine Edge-optimierte API um:

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
    --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Da es put-rest-apisich um die Aktualisierung von API-Definitionen handelt, ist sie nicht für die Aktualisierung eines API-Endpunkttyps geeignet.

Methoden für REST-APIs in API Gateway

In API Gateway setzt sich eine API-Methode aus einer Methodenanfrage und eine Methodenantwort zusammen. Sie richten eine API-Methode ein, um zu definieren, was ein Client tun sollte oder tun muss, um eine Anforderung für den Zugriff auf den Service am Backend zu übermitteln und um die Antworten zu definieren, die der Client im Gegenzug erhält. Für die Eingabe können Sie Methodenanforderungs-Parameter oder eine anwendbare Nutzlast für den Client auswählen, um die erforderlichen oder optionalen Daten zur Laufzeit bereitzustellen. Für die Ausgabe bestimmen Sie den Statuscode der Methodenantwort, die Header und den geltenden Textkörper als Ziele denen Backend-Antwortdaten zugewiesen werden, bevor die Rückgabe an den Client erfolgt. Um den Client-Developer dabei zu unterstützen, das Verhalten und die Eingabe- und Ausgabeformate Ihrer API zu verstehen, können Sie Ihre API dokumentieren und ordnungsgemäße Fehlermeldungen für ungültige Anforderungen bereitstellen.

Eine API-Methodenanforderung ist eine HTTP-Anforderung. Um die Methodenanforderung einzurichten, konfigurieren Sie eine HTTP-Methode (oder einVerb), den Pfad zu einer API-Ressource, Header, geltende Parameter für Abfragezeichenfolgen. Außerdem konfigurieren Sie eine Nutzlast, wenn die HTTP-Methode `POST`, `PUT`oder `PATCH` ist. Um beispielsweise mithilfe der PetStore Beispiel-API ein Haustier abzurufen, definieren Sie die API-Methodenanforderung von. `GET /pets/{petId}` Dabei `{petId}` handelt es sich um einen Pfadparameter, der zur Laufzeit eine Zahl annehmen kann.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
```

```
...
```

Wenn der Client einen falschen Pfad angibt, zum Beispiel `/pet/1` oder `/pets/one` anstatt `/pets/1`, wird eine Ausnahme ausgelöst.

Eine API-Methodenantwort ist eine HTTP-Antwort mit einem bestimmten Statuscode. Für eine Nicht-Proxy-Integration müssen Sie Methodenantworten einrichten, um die erforderlichen oder optionalen Ziele von Mappings anzugeben. Diese transformieren Integrationsantwort-Header oder einen Textkörper in verknüpfte Methodenantwort-Header oder einen Textkörper. Das Mapping kann so einfach wie eine identische Transformation sein, mit der die Header oder der Text unverändert mittels die Integration übergeben werden. Zum Beispiel zeigt die folgende `200`-Methodenantwort ein Beispiel für die Weiterleitung einer unveränderten erfolgreichen Integrationsantwort.

```
200 OK
Content-Type: application/json
...

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

In der Regel können Sie eine Methodenantwort entsprechend einer bestimmten Antwort vom Backend definieren. In der Regel handelt es sich dabei um alle 2XX-, 4XX- und 5XX-Antworten. Dies kann jedoch nicht praktikabel sein, da Sie häufig nicht alle Antworten, die ein Backend möglicherweise zurückgibt, im Vorfeld wissen können. In der Praxis können Sie eine Methodenantwort als Standard festlegen, um die unbekannten oder nicht zugewiesenen Antworten vom Backend zu verarbeiten. Es ist ein bewährtes Verfahren, die 500-Antwort als Standard festzulegen. In jedem Fall müssen Sie mindestens eine Methodenantwort für Nicht-Proxy-Integrationen einrichten. Andernfalls gibt API Gateway eine 500-Fehler-Antwort an den Client zurück, selbst wenn die Anfrage am Backend erfolgreich ist.

Zur Unterstützung eines stark typisierten SDKs, z. B. ein Java-SDK für Ihre API, sollten Sie das Datenmodell zur Eingabe für die Methodenanforderung und das Datenmodell zur Ausgabe der Methodenantwort definieren.

Voraussetzungen

Vor dem Einrichten einer API-Methode, überprüfen Sie Folgendes:

- Die Methode muss in API Gateway verfügbar sein. Folgen Sie den Anweisungen in Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen.
- Wenn die Methode mit einer Lambda-Funktion kommunizieren soll, müssen Sie die Lambda-Aufrufrolle und die Lambda-Ausführungsrolle bereits in IAM erstellt haben. Sie müssen außerdem die Lambda-Funktion erstellt haben, mit der Ihre Methode in AWS Lambda kommuniziert. Befolgen Sie zum Erstellen der Rollen und der Funktion die Anweisungen in Lambda-Funktion für die Nicht-Proxy-Integration von Lambda erstellen unter Wählen Sie ein AWS Lambda Integrations-Tutorial.
- Wenn Sie möchten, dass die Methode mit einer HTTP- oder HTTP-Proxy-Integration kommunizieren soll, muss die HTTP-Endpunkt-URL, mit der die Methode kommuniziert, bereits erstellt sein und Sie müssen Zugriff darauf haben.
- Stellen Sie sicher, dass Ihre Zertifikate für HTTP- und HTTP-Proxy-Endpunkte von API Gateway unterstützt werden. Weitere Informationen finden Sie unter Von API Gateway unterstützte Autorisierungen für HTTP- und HTTP-Proxy-Integrationen.

> **Note**
>
> Wenn Sie mit der REST-API-Konsole eine Methode erstellen, konfigurieren Sie sowohl die Integrationsanforderung als auch die Methodenanforderung. Weitere Informationen finden Sie unter the section called “ Einrichten einer API-Integrationsanforderung mit der Konsole”.

Themen

Methodenanfrage in API Gateway einrichten

Zum Einrichten einer Methodenanforderung müssen nach dem Erstellen einer RestApiRessource die folgenden Aufgaben ausgeführt werden:

1. Erstellen einer neuen API oder Auswahl einer vorhandenen API-Ressourcen-Entität.
2. Erstellen einer API-Methoden-Ressource, die ein spezifisches HTTP-Verb für die neue oder ausgewählte API-`Resource` ist. Diese Aufgabe kann noch weiter in die folgenden Sub-Aufgaben unterteilt werden:

- Hinzufügen einer HTTP-Methode zur Methodenanforderung
- Konfigurieren von Anforderungsparametern
- Definieren eines Modells für den Anforderungstextkörper
- Einfügen eines Autorisierungsschemas
- Aktivieren einer Anforderungsvalidierung

Sie können diese Aufgaben mithilfe der folgenden Methoden durchführen:

- API Gateway-Konsole
- AWS CLI Befehle (create-resource und put-method)
- AWS SDK-Funktionen (z. B. in Node.js, createResource und PutMethod)
- API Gateway-REST-API (resource:create und method:put).

Themen

- Einrichten von API-Ressourcen
- Einrichten einer HTTP-Methode
- Einrichten von Methodenanforderungs-Parametern
- Einrichten des Modells der Methodenanforderung
- Einrichten der Autorisierung der Methodenanforderung
- Einrichten einer Validierung der Methodenanforderung

Einrichten von API-Ressourcen

In einer API Gateway-API setzen Sie adressierbare Ressourcen als eine API-Ressourcen-Entitätsstruktur ein, mit der Stammressource (/) oben in der Hierarchie. Die Stammressource bezieht sich auf die Basis-URL der API, die aus dem API-Endpunkt und einem Stufennamen besteht. In der API Gateway-Konsole wird diese Basis-URI als Invoke URI (Aufruf-URI) bezeichnet und im Stage-Editor der API angezeigt, nachdem die API bereitgestellt wurde.

Der API-Endpunkt kann ein Standard-Host-Name oder ein benutzerdefinierter Domänenname sein. Der Standard-Host-Name hat das folgende Format:

```
{api-id}.execute-api.{region}.amazonaws.com
```

In diesem Format stellt der *{api-id}* die API-ID dar, die von API Gateway generiert wird. Die *{region}*-Variable stellt die AWS -Region (z. B. `us-east-1`) dar, die Sie bei der Erstellung der API auswählen. Ein benutzerdefinierter Domänenname ist jeder benutzerfreundliche Name in einer gültigen Internetdomäne. Wenn Sie beispielsweise eine Internetdomäne von `example.com` registriert haben, ist alles von `*.example.com` ein gültiger benutzerdefinierter Domänenname. Weitere Informationen finden Sie unter Erstellen eines benutzerdefinierten Domänenname.

Für die PetStore Beispiel-API macht die Root-Ressource (`/`) die Tierhandlung verfügbar. Die `/pets`-Ressource stellt die Sammlung von Haustieren, die im PetStore verfügbar sind, dar. Die `/pets/{petId}` stellt ein einzelnes Haustier einer bestimmten ID (`petId`) dar. Der Pfadparameter von `{petId}` ist Teil der Anforderungsparameter.

Zum Konfigurieren einer API-Ressource wählen Sie eine vorhandene Ressource als übergeordnetes Element aus und erstellen anschließend unter dieser die untergeordnete Ressource. Sie beginnen mit der Stammressource als übergeordnetes Element, fügen diesem eine Ressource hinzu, fügen eine andere Ressource dieser untergeordneten Ressource als neues übergeordnetes Element und so weiter der übergeordneten ID hinzu. Anschließend fügen Sie die benannte Ressource dem übergeordneten Element hinzu.

Mit können Sie den `get-resources` Befehl aufrufen AWS CLI, um herauszufinden, welche Ressourcen einer API verfügbar sind:

```
aws apigateway get-resources --rest-api-id <apiId> \
                             --region <region>
```

Das Ergebnis ist eine Liste der aktuell verfügbaren Ressourcen der API. Für unsere PetStore Beispiel-API sieht diese Liste wie folgt aus:

```
{
    "items": [
        {
            "path": "/pets",
            "resourceMethods": {
                "GET": {}
            },
            "id": "6sxz2j",
            "pathPart": "pets",
            "parentId": "svzr2028x8"
        },
        {
```

```
            "path": "/pets/{petId}",
            "resourceMethods": {
                "GET": {}
            },
            "id": "rjkmth",
            "pathPart": "{petId}",
            "parentId": "6sxz2j"
        },
        {
            "path": "/",
            "id": "svzr2028x8"
        }
    ]
}
```

Jedes Element listet die ID der Ressource (`id`) und, mit Ausnahme der Stammressource, das unmittelbar übergeordnete Element (`parentId`) sowie den Ressourcennamen (`pathPart`). Die Stammressource ist dahingehend besonders, dass sie über kein übergeordnetes Element verfügt. Nachdem Sie eine Ressource als übergeordnetes Element auswählt haben, rufen Sie den folgenden Befehl auf, um eine untergeordnete Ressource hinzuzufügen.

```
aws apigateway create-resource --rest-api-id <apiId> \
                               --region <region> \
                               --parent-id <parentId> \
                               --path-part <resourceName>
```

Um beispielsweise Tiernahrung zum Verkauf auf der PetStore Website hinzuzufügen, fügen Sie dem Stamm (/) eine `food` Ressource hinzu, indem Sie `path-part` auf `food` und `parent-id` to setzen`svzr2028x8`. Das Ergebnis sollte wie folgt aussehen:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Verwenden Sie eine Proxy-Ressource zur Optimierung der API-Einrichtung

Wenn das Geschäft wächst, kann der PetStore Eigentümer beschließen, Lebensmittel, Spielzeug und andere Artikel für Haustiere zum Verkauf anzubieten. Um dies zu unterstützen, können

Sie `/food`, `/toys` und andere Ressourcen unter der Stammressource hinzufügen. Unter jeder Verkaufskategorie möchten Sie möglicherweise auch weitere Ressourcen hinzufügen, wie `/food/{type}/{item}`, `/toys/{type}/{item}` usw. Dies kann sehr zeitaufwändig werden. Wenn Sie sich dazu entscheiden, eine mittlere Schicht `{subtype}` den Ressourcenpfaden hinzuzufügen, um die Pfadhierarchie in `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`usw. zu ändern, könnten die Änderungen zu Fehlern bei der vorhandenen API-Einrichtung führen. Um dies zu vermeiden, können Sie ein API Gateway-Proxy-Ressource verwenden, um einen Satz von API-Ressourcen auf einmal bereitzustellen.

API Gateway definiert eine Proxy-Ressource als Platzhalter für eine Ressource, die bei der Übertragung der Anfrage angegeben wird. Eine Proxy-Ressource wird durch einen speziellen Pfadparameter `{proxy+}` ausgedrückt, häufig auch als gieriger Pfadparameter bezeichnet. Das +-Zeichen gibt an, welche untergeordneten Ressourcen ihm angehängt werden. Der Platzhalter `/parent/{proxy+}` steht für jede Ressource, die mit dem Pfadmuster `/parent/*` übereinstimmt. Der Name des gierigen Pfadparameters, `proxy`, kann auf die gleichen Art und Weise wie der Name eines regulären Pfadparameters durch eine andere Zeichenfolge ersetzt werden.

Mit dem rufen Sie den folgenden Befehl auf AWS CLI, um eine Proxyressource unter dem Stammverzeichnis (`/{proxy+}`) einzurichten:

```
aws apigateway create-resource --rest-api-id <apiId> \
                               --region <region> \
                               --parent-id <rootResourceId> \
                               --path-part {proxy+}
```

Das Ergebnis sollte wie folgt aussehen:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "234jdr",
    "parentId": "svzr2028x8"
}
```

Für das `PetStore`-API-Beispiel können Sie `/{proxy+}` verwenden, um die `/pets` und `/pets/{petId}` darzustellen. Diese Proxyressource kann auch auf alle anderen (vorhandenen oder to-be-added) Ressourcen verweisen `/food/{type}/{item}/toys/{type}/{item}`, z. B. auf, usw. oder `/food/{type}/{subtype}/{item}/toys/{type}/{subtype}/{item}`, usw. Die Backend-Developer bestimmt die Ressourcenhierarchie und der Client-Developer ist dafür

verantwortlich, sie zu verstehen. API Gateway übergibt einfach alles, was der Client an das Backend übermittelt hat.

Eine API kann über mehr als eine Proxy-Ressource verfügen. Beispielsweise sind die folgenden Proxy-Ressourcen innerhalb einer API erlaubt.

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Wenn eine Proxy-Ressource über gleichwertige Nicht-Proxy-Elemente verfügt, werden die gleichwertigen Ressourcen von der Darstellung der Proxy-Ressource ausgeschlossen. Bei den vorherigen Beispielen bezieht sich `/{proxy+}` auf alle Ressourcen unter der Stammressource mit Ausnahme der `/parent[/*]`-Ressourcen. Mit anderen Worten, eine Methodenanforderung für eine bestimmte Ressource hat Vorrang vor einer ähnlichen Methodenanforderung für eine generische Ressource auf derselben Ebene der Ressourcenhierarchie.

Eine Proxy-Ressource kann über keine untergeordnete Ressource verfügen. Jede API-Ressource nach `{proxy+}` ist überflüssig und zweideutig. Die folgenden Proxy-Ressourcen sind nicht innerhalb einer API erlaubt.

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Einrichten einer HTTP-Methode

Eine API-Methodenanforderung wird durch die API Gateway-Methode-Ressource gekapselt. Um die Methodenanforderung einzurichten, müssen Sie zuerst die `Method`-Ressource instanziieren und mindestens eine HTTP-Methode und einen Autorisierungstyp für die Methode festlegen.

API Gateway ist eng mit der Proxy-Ressource verbunden und unterstützt die HTTP-API-Methode `ANY`. Diese `ANY`-Methode entspricht der HTTP-Methode, die zur Laufzeit bereitgestellt wird. Sie gibt Ihnen die Möglichkeit, eine einzige API-Methodeneinrichtung für die unterstützten HTTP-Methoden `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` und `PUT` zu verwenden.

Sie können die `ANY`-Methode auch auf einer Nicht-Proxy-Ressource einrichten. Durch die Kombination der `ANY`-Methode mit einer Proxy-Ressource erhalten Sie eine einzige API-Methodeneinrichtung für alle unterstützten HTTP-Methoden gegen alle Ressourcen einer API.

Darüber hinaus kann sich das Backend weiterentwickeln, ohne die vorhandene API-Einrichtung zu stören.

Berücksichtigen Sie vor dem Einrichten einer API-Methode, wer die Methode aufrufen kann. Legen Sie den Autorisierungstyp entsprechend Ihres Plans fest. Für einen offenen Zugriff setzen Sie ihn auf `NONE`. Zur Nutzung von IAM-Berechtigungen setzen Sie den Autorisierungstyp auf `AWS_IAM`. Um eine Lambda-Genehmiger-Funktion zu verwenden, setzen Sie diese Eigenschaft auf `CUSTOM`. Um einen Amazon Cognito-Benutzerpool zu verwenden, setzen Sie den Autorisierungstyp auf `COGNITO_USER_POOLS`.

Der folgende AWS CLI Befehl zeigt, wie eine Methodenanforderung des `ANY` Verbs für eine angegebene Ressource (`6sxz2j`) erstellt wird, wobei die IAM-Berechtigungen zur Steuerung des Zugriffs verwendet werden.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method ANY \
       --authorization-type AWS_IAM \
       --region us-west-2
```

Informationen zur Erstellung einer API-Methodenanforderung mit einem anderen Autorisierungstyp finden Sie unter the section called “Einrichten der Autorisierung der Methodenanforderung”.

Einrichten von Methodenanforderungs-Parametern

Methodenanforderungs-Parameter sind eine Möglichkeit für einen Client, Eingabedaten oder Ausführungskontext bereitzustellen, damit die Methodenanforderung abgeschlossen werden kann. Ein Methodenparameter kann ein Pfadparameter, ein Header oder ein Abfragezeichenfolge-Parameter sein. Im Rahmen der Methodenanforderungseinrichtung müssen Sie die erforderlichen Anforderungsparameter deklarieren, um sie dem Client zur Verfügung zu stellen. Für eine Nicht-Proxy-Integration können Sie diese Anforderungsparameter in ein Formular umwandeln, das mit der Backend-Anforderung kompatibel ist.

Zum Beispiel ist für die `GET /pets/{petId}`-Methodenanforderung die `{petId}`-Pfadvariable ein erforderlicher Anforderungsparameter. Sie können diesen Pfadparameter beim Aufrufen des Befehls AWS CLI der `put-method` deklarieren. Dies kann wie folgt dargestellt werden:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id rjkmth \
       --http-method GET \
```

```
        --authorization-type "NONE" \
        --region us-west-2 \
        --request-parameters method.request.path.petId=true
```

Wenn ein Parameter nicht erforderlich ist, können Sie ihn auf `false` in `request-parameters` setzen. Wenn beispielsweise die `GET /pets`-Methode den optionalen Abfragezeichenfolge-Parameter `type` und den optionalen Header-Parameter `breed` verwendet, können Sie diese mithilfe des folgenden CLI-Befehls deklarieren, vorausgesetzt, dass die `/pets`-Ressourcen-`id` `6sxz2j` lautet:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --authorization-type "NONE" \
        --region us-west-2 \
        --request-parameters
 method.request.querystring.type=false,method.request.header.breed=false
```

Anstelle dieser Kurzform können Sie eine JSON-Zeichenfolge verwenden, um den `request-parameters`-Wert zu setzen:

```
'{"method.request.querystring.type":false,"method.request.header.breed":false}'
```

Mit dieser Einrichtung kann der Client die Haustiere nach Typ abfragen:

```
GET /pets?type=dog
```

Und der Client kann Hunde der Rasse Pudel wie folgt abfragen:

```
GET /pets?type=dog
breed:poodle
```

Weitere Informationen, wie Methodenanforderungs-Parameter den Integrationsanforderungs-Parametern zugewiesen werden, finden Sie unter the section called “Integrationen”.

Einrichten des Modells der Methodenanforderung

Für eine API-Methode, die Eingabedaten in eine Nutzlast übernehmen kann, können Sie ein Modell verwenden. Ein Modell wird in einem JSON-Schema Entwurf 4 ausgedrückt und beschreibt die Datenstruktur des Anforderungstexts. Mit einem Modell kann ein Client festlegen, wie eine

Methodenanforderungsnutzlast als Eingabe erstellt wird. Noch wichtiger ist, dass API Gateway das Modell verwendet, um eine Anfrage zu validieren, ein SDK zu generieren und eine Mapping-Vorlage für die Einrichtung der Integration in der API Gateway-Konsole zu initialisieren. Informationen zum Erstellen eines Modells finden Sie unter Grundlegendes zu Datenmodellen.

Je nach Inhaltstypen, kann eine Methodenutzlast über unterschiedliche Formate verfügen. Ein Modell wird für den Medientyp der angewendeten Nutzlast indiziert. API Gateway verwendet den `Content-Type` Anforderungsheader, um den Inhaltstyp zu bestimmen. Um Methodenanforderungsmodelle einzurichten, fügen Sie der `requestModels` Map beim Aufrufen des Befehls Schlüssel-Wert-Paare des "*<media-type>*":"*<model-name>*" Formats hinzu. AWS CLI `put-method`

Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

Um beispielsweise ein Modell für die JSON-Nutzlast der `POST /pets` Methodenanforderung der PetStore Beispiel-API festzulegen, können Sie den folgenden Befehl aufrufen: AWS CLI

```
aws apigateway put-method \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method POST \
       --authorization-type "NONE" \
       --region us-west-2 \
       --request-models '{"application/json":"petModel"}'
```

Hier ist `petModel` der `name`-Eigenschaftswert einer `Model`-Ressource, die ein Haustier beschreibt. Die tatsächliche Schemadefinition wird als JSON-Zeichenfolgenwert der `schema`-Eigenschaft der `Model`-Ressource ausgedrückt.

In einem Java oder anderen stark typisierten SDK der API werden die Eingabedaten in die `petModel`-Klasse umgewandelt, die von der Schemadefinition abgeleitet sind. Mit dem Anforderungsmodell werden die Eingabedaten in dem generierten SDK in die `Empty`-Klasse umgewandelt, die von dem Standard-`Empty`-Modell abgeleitet wird. In diesem Fall kann der Client nicht die richtige Datenklasse instanziieren, um die benötigte Ausgabe bereitzustellen.

Einrichten der Autorisierung der Methodenanforderung

Um zu steuern, wer die API-Methode aufrufen kann, können Sie den Autorisierungstyp für die Methode konfigurieren. Sie können diesen Typ verwenden, um einen der unterstützten Genehmiger

einzusetzen, einschließlich IAM-Rollen und -Richtlinien (`AWS_IAM`), einen Amazon Cognito-Benutzerpool (`COGNITO_USER_POOLS`) oder einen Lambda-Genehmiger (`CUSTOM`).

Um IAM-Berechtigungen für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf **`AWS_IAM`**. Wenn Sie diese Option festlegen, verifiziert API Gateway die Signatur des Aufrufers in der Anfrage basierend auf den Anmeldeinformationen des Aufrufers. Wenn der bestätigte Benutzer über die Berechtigung zum Aufrufen der Methode verfügt, akzeptiert sie die Anforderung. Andernfalls lehnt sie die Anforderung ab und der Aufrufer erhält die Fehlerantwort „Unbefugt“. Der Aufruf der Methode ist nicht erfolgreich, es sei denn, der Aufrufer hat die Berechtigung, die API-Methode aufzurufen. Die folgende IAM-Richtlinie gewährt dem Aufrufer die Berechtigung, alle darin API-Methoden aufzurufen, die im gleichen AWS-Konto erstellt wurden:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": "arn:aws:execute-api:*:*:*"
        }
    ]
}
```

Weitere Informationen finden Sie unter the section called “Verwenden von IAM-Berechtigungen”.

Derzeit können Sie diese Richtlinie nur Benutzern, Gruppen und Rollen innerhalb des AWS-Konto des API-Besitzers gewähren. Benutzer aus einem anderen Land AWS-Konto können die API-Methoden nur aufrufen, wenn sie eine Rolle innerhalb des API-Besitzers AWS-Konto übernehmen dürfen und über die erforderlichen Berechtigungen verfügen, um die `execute-api:Invoke` Aktion aufzurufen. Weitere Informationen zu kontoübergreifenden Berechtigungen finden Sie unter Verwenden von IAM-Rollen.

Sie können ein AWS SDK oder einen REST-API-Client wie Postman verwenden AWS CLI, der Signature Version 4 (Sigv4) -Signaturen implementiert.

Um einen Lambda-Genehmiger für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf `CUSTOM` und die `authorizer-id`-

Eingabeeigenschaft auf den `id`-Eigenschaftswert eines Lambda-Genehmigers, der bereits vorhanden ist. Der referenzierte Lambda-Genehmiger kann vom Typ `TOKEN` oder `REQUEST` sein. Informationen zur Erstellung eines Lambda-Genehmigers finden Sie unter the section called “Lambda-Genehmiger verwenden”.

Um einen Amazon Cognito-Benutzerpool für die Autorisierung des Zugriffs auf die API-Methode zu verwenden, setzen Sie die `authorization-type`-Eingabeeigenschaft auf `COGNITO_USER_POOLS` und die `authorizer-id`-Eingabeeigenschaft auf den `id`-Eigenschaftswert des `COGNITO_USER_POOLS`-Genehmigers, der bereits vorhanden ist. Informationen über die Erstellung eines Genehmigers für einen Amazon Cognito-Benutzerpool finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”.

Einrichten einer Validierung der Methodenanforderung

Sie können die Anforderungsvalidierung beim Einrichten einer API-Methodenanforderung aktivieren. Sie müssen zuerst eine Anforderungsvalidierung erstellen:

```
aws apigateway create-request-validator \
    --rest-api-id 7zw9uyk9kl \
    --name bodyOnlyValidator \
    --validate-request-body  \
    --no-validate-request-parameters
```

Dieser CLI-Befehl erstellt eine Nur-Text-Anforderungsvalidierung. Die Beispielausgabe lautet wie folgt:

```
{
    "validateRequestParameters": false,
    "validateRequestBody": true,
    "id": "jgpyy6",
    "name": "bodyOnlyValidator"
}
```

Mit dieser Anforderungsvalidierung können Sie Anforderungsvalidierung als Teil der Einrichtung der Methodenanforderung aktivieren:

```
aws apigateway put-method \
    --rest-api-id 7zw9uyk9kl
    --region us-west-2
```

```
    --resource-id xdsvhp
    --http-method PUT
    --authorization-type "NONE"
    --request-parameters '{"method.request.querystring.type": false,
"method.request.querystring.page":false}'
    --request-models '{"application/json":"petModel"}'
    --request-validator-id jgpyy6
```

Um in einer Anforderungsvalidierung aufgenommen zu werden, muss ein Anforderungsparameter als erforderlich deklariert werden. Wenn der Abfragezeichenfolge-Parameter für die Seite in einer Anforderungsvalidierung verwendet wird, muss die `request-parameters`-Map des vorangehenden Beispiels als `'{"method.request.querystring.type": false, "method.request.querystring.page":true}'` festgelegt werden.

Methodenantworten in API Gateway einrichten

Eine API-Methodenantwort kapselt die Ausgabe einer API-Methodenanforderung, die der Client erhält. Die Ausgabedaten enthalten einen HTTP-Statuscode, einige Header und in der Regel einen Textkörper.

Mit Nicht-Proxy-Integrationen können die angegebenen Antwortparameter und Textkörper von den zugeordneten Integrationsantwortdaten oder bestimmten statischen Werten entsprechend der Mappings zugewiesen werden. Diese Mappings werden in der Integrationsantwort beschrieben. Das Mapping kann eine identische Transformation sein, die die Integrationsantwort unverändert weiterleitet.

Bei einer Proxy-Integration leitet API Gateway die Antwort des Backends automatisch an die Antwort der Methode weiter. Sie müssen sich nicht um die Einrichtung der API-Methodenantwort kümmern. Bei der Lambda-Proxy-Integration muss die Lambda-Funktion jedoch ein Ergebnis in diesem Ausgabeformat zurückgeben, damit API Gateway die Integrationsantwort erfolgreich einer Methodenantwort zuordnen kann.

Programmatisch gesehen läuft die Einrichtung der Methodenantwort darauf hinaus, eine MethodResponseRessource von API Gateway zu erstellen und die Eigenschaften von statusCode, ResponseParameters und ResponseModels festzulegen.

Wenn Sie Statuscodes für eine API-Methode festlegen, sollten Sie einen als Standard auswählen, um jede Integrationsantwort eines unerwarteten Statuscodes zu bearbeiten. Es ist sinnvoll, `500` als Standard festzulegen, da dies mit dem Umwandeln von ansonsten nicht zugeordneten Antworten als

serverseitiger Fehler gleichzusetzen ist. Zur Vereinfachung legt die API Gateway-Konsole die `200`-Antwort als Standard fest. Sie können dies jedoch auf die `500`-Antwort zurücksetzen.

Um eine Methodenantwort einzurichten, müssen Sie die Methodenanforderung erstellt haben.

Einrichten des Statuscodes der Methodenantwort

Der Statuscode einer Methodenantwort definiert einen Antworttyp. Beispiel: Antworten von 200, 400 und 500 geben jeweils erfolgreiche, clientseitige Fehler und serverseitige Fehlerantworten an.

Um den Statuscode einer Methodenantwort einzurichten, legen Sie die `statusCode`-Eigenschaft auf einen HTTP-Statuscode fest. Der folgende AWS CLI -Befehl erstellt eine Methodenantwort von `200`.

```
aws apigateway put-method-response \
       --region us-west-2  \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method GET \
       --status-code 200
```

Einrichten von Parametern der Methodenantwort

Parameter der Methodenantwort definieren, welche Header der Client als Reaktion auf die zugeordnete Methodenanforderung erhält. Antwortparameter legen auch ein Ziel fest, dem API Gateway einen Parameter der Integrationsantwort entsprechend der Mappings zuweist, die in der Integrationsantwort der API beschrieben sind.

Um die Parameter der Methodenantwort einzurichten, fügen Sie der `responseParameters`-Zuordnung der `MethodResponse`-Schlüssel-Wert-Paare im Format `"{parameter-name}":"{boolean}"` hinzu. Der folgende CLI-Befehl zeigt ein Beispiel für die Einstellung des `my-header` Headers.

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false
```

Einrichten von Modellen der Methodenantwort

Ein Modell der Methodenantwort definiert ein Format des Textkörpers der Methodenantwort. Bevor Sie das Antwortmodell einrichten können, müssen Sie das Modell zunächst in API Gateway erstellen. Dazu können Sie den Befehl `create-model` aufrufen. Das folgende Beispiel zeigt, wie Sie ein `PetStorePet`-Modell erstellen, um den Textkörper der Antwort der `GET /pets/{petId}`-Methodenanforderung zu beschreiben.

```
aws apigateway create-model \
    --region us-west-2 \
    --rest-api-id vaz7da96z6 \
    --content-type application/json \
    --name PetStorePet \
    --schema '{ \
                "$schema": "http://json-schema.org/draft-04/schema#", \
                "title": "PetStorePet", \
                "type": "object", \
                "properties": { \
                  "id": { "type": "number" }, \
                  "type": { "type": "string" }, \
                  "price": { "type": "number" } \
                } \
             }'
```

Das Ergebnis wird als eine API Gateway-`Model`-Ressource erstellt.

Um die Methodenantwortmodelle zur Definition des Payload-Formats einzurichten, fügen Sie der Ressourcenzuweisung das Schlüssel-Wert-Paar „application/json“:“ PetStorePet "hinzu. `requestModels`MethodResponse Der folgende AWS CLI Befehl von `put-method-response` zeigt, wie das gemacht wird:

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false \
        --response-models '{"application/json":"PetStorePet"}'
```

Das Einrichten eines Methodenantwort-Modells ist erforderlich, wenn Sie einen stark typisierten SDK für die API generieren. Es stellt sicher, dass die Ausgabe an eine geeignete Klasse in Java oder Objective-C übergeben wird. In anderen Fällen ist das Einrichten eines Modells optional.

Methode über die API Gateway-Konsole einrichten

Wenn Sie mit der REST-API-Konsole eine Methode erstellen, konfigurieren Sie sowohl die Integrationsanforderung als auch die Methodenanforderung. Standardmäßig erstellt API Gateway die `200` Methodenantwort für Ihre Methode.

Die folgenden Anweisungen zeigen, wie Sie die Einstellungen für Methodenanfragen bearbeiten und zusätzliche Methodenantworten für Ihre Methode erstellen.

Themen

Bearbeiten Sie eine API Gateway Gateway-Methodenanforderung in der API Gateway Gateway-Konsole

Bei diesen Anweisungen wird davon ausgegangen, dass Sie Ihre Methodenanforderung bereits erstellt haben. Weitere Informationen zum Erstellen einer Methode finden Sie unterthe section called “Einrichten einer API-Integrationsanforderung mit der Konsole”.

1. Wählen Sie im Bereich Ressourcen Ihre Methode und dann die Registerkarte Methodenanfrage aus.
2. Wählen Sie im Abschnitt Method request settings (Einstellungen der Methodenanforderung) die Option Edit (Bearbeiten) aus.
3. Wählen Sie für Authorization (Autorisierung) einen verfügbaren Genehmiger aus.

    a. Um einen offenen Zugriff auf die Methode für alle Benutzer zu aktivieren, wählen Sie None (Keine) aus. Dieser Schritt kann übersprungen werden, wenn die Standardeinstellung nicht geändert wurde.

    b. Um IAM-Berechtigungen für die Steuerung des Client-Zugriffs auf die Methode zu verwenden, wählen Sie `AWS_IAM` aus. Mit dieser Auswahl können nur Benutzer der IAM-Rollen mit der korrekten angefügten IAM-Richtlinie diese Methode aufrufen.

    Legen Sie zum Erstellen der IAM-Rolle eine Zugriffsrichtlinie mit einem Format wie folgt fest:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "resource-statement"
      ]
    }
  ]
}
```

In dieser Zugriffsrichtlinie ist *Resource-Statement* der ARN Ihrer Methode. Sie finden den ARN Ihrer Methode, indem Sie die Methode auf der Seite Ressourcen auswählen. Weitere Informationen zum Festlegen der IAM-Berechtigungen finden Sie unter Kontrollieren des Zugriffs auf eine API mit IAM-Berechtigungen.

Um die IAM-Rolle zu erstellen, können Sie die Anweisungen im folgenden Tutorial anpassen. ???

c. Um einen Lambda-Genehmiger zu verwenden, wählen Sie ein Token oder einen Genehmiger der Anforderung aus. Erstellen Sie einen Lambda-Genehmiger, damit diese Auswahl im Dropdown-Menü angezeigt wird. Informationen über die Erstellung eines Lambda-Genehmigers finden Sie unter API Gateway-Lambda-Genehmiger verwenden.

d. Um einen Amazon Cognito-Benutzerpool zu verwenden, wählen Sie einen verfügbaren Benutzerpool unter Cognito user pool authorizers (Cognito-Benutzerpool-Genehmiger) aus. Erstellen Sie einen Benutzerpool in Amazon Cognito und einen Amazon Cognito-Benutzerpool-Genehmiger in API Gateway, damit diese Auswahl im Dropdown-Menü angezeigt wird. Informationen darüber, wie Sie einen Amazon Cognito-Benutzerpool-Genehmiger erstellen, finden Sie unter Zugriff auf eine REST-API mit Amazon Cognito-Benutzerpools als Genehmiger steuern.

4. Um die Anforderungsvalidierung anzugeben, wählen Sie einen Wert aus dem Dropdown-Menü Request Validator (Anforderungs-Validator) aus. Um die Anforderungsvalidierung zu deaktivieren, wählen Sie None (Keine) aus. Weitere Informationen zu den jeweiligen Optionen finden Sie unter Verwenden der Anforderungsvalidierung in API Gateway.

5. Wählen Sie API key required (API-Schlüssel erforderlich) aus, damit ein API-Schlüssel erforderlich ist. Wenn diese Option aktiviert ist, werden API-Schlüssel in Nutzungsplänen verwendet, um den Client-Datenverkehr zu drosseln.
6. (Optional) Um einen Operationsnamen in einem Java SDK dieser API zuzuweisen, die von einem API Gateway generiert wird, geben Sie unter Operation name (Operationsname) einen Namen ein. Beispielsweise ist für die Methodenanforderung `GET /pets/{petId}` der entsprechende Java-SDK-Operationsname standardmäßig `GetPetsPetId`. Dieser Name wird aus dem HTTP-Verb (`GET`) der Methode und den Variablennamen des Ressourcenpfads (`Pets` und `PetId`) erstellt. Wenn Sie den Operationsnamen als `getPetById` festlegen, wird der SDK-Operationsnamen zu `GetPetById`.
7. Fügen Sie der Methode wie folgt einen Abfragezeichenfolge-Parameter hinzu:

   a. Wählen Sie URL Query String Parameters (URL-Abfragezeichenfolgen-Parameter) und dann Add query string (Abfragezeichenfolge hinzufügen) aus.

   b. Geben Sie unter Name den Namen des Abfragezeichenfolge-Parameters ein.

   c. Wenn der neu erstellte Abfragezeichenfolge-Parameter für die Anforderungsvalidierung verwendet wird, wählen Sie die Option Required (Obligatorisch) aus. Weitere Informationen zur Anforderungsvalidierung finden Sie unter Verwenden der Anforderungsvalidierung in API Gateway.

   d. Wenn der neu erstellte Abfragezeichenfolge-Parameter als Teil eines Caching-Schlüssels verwendet wird, wählen Sie die Option Caching aus. Weitere Informationen zum Caching finden Sie unter Verwenden der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren.

   Klicken Sie zum Entfernen des Abfragezeichenfolge-Parameters auf Remove (Entfernen).
8. Fügen Sie der Methode wie folgt einen Header-Parameter hinzu:

   a. Wählen Sie HTTP Request Headers (HTTP-Anforderungs-Header) und dann Add header (Header hinzufügen) aus.

   b. Geben Sie unter Name den Namen des Headers ein.

   c. Wenn der neu erstellte Header für die Anforderungsvalidierung verwendet wird, wählen Sie die Option Required (Obligatorisch) aus. Weitere Informationen zur Anforderungsvalidierung finden Sie unter Verwenden der Anforderungsvalidierung in API Gateway.

   d. Wenn der neu erstellte Hader als Teil eines Caching-Schlüssels verwendet wird, wählen Sie die Option Caching aus. Weitere Informationen zum Caching finden Sie unter Verwenden

der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren.

Um de Header zu entfernen, wählen Sie Remove (Entfernen).

9. Um das Nutzlastformat einer Methodenanforderung mit dem `POST`-, `PUT`- oder `PATCH`-HTTP-Verb zu deklarieren, erweitern Sie Request Body (Anforderungstext) und führen Sie die folgenden Schritte aus:

   a. Wählen Sie Add model aus.

   b. Geben Sie unter Content-type einen MIME-Typ (beispielsweise `application/json`) ein.

   c. Wählen Sie für Modell ein Modell aus dem Dropdown-Menü aus. Die derzeit verfügbaren Modelle für die API umfassen die standardmäßigen `Empty`- und `Error`-Modelle sowie alle Modelle, die Sie erstellt und der Modell-Sammlung der API hinzugefügt haben. Weitere Informationen zum Erstellen eines Modells finden Sie unter Datenmodelle.

   > **Note**
   >
   > Das Modell ist nützlich, um den Client über das erwartete Datenformat einer Nutzlast zu informieren. Es ist hilfreich, ein Mapping-Vorlagenskelett zu generieren. Es ist wichtig, einen stark typisierten SDK der API in solchen Sprachen wie Java, C#, Objective-C und Swift zu generieren. Es ist nur erforderlich, wenn die Anforderungsvalidierung für die Nutzlast aktiviert ist.

10. Wählen Sie Speichern.

API Gateway-Methodenantwort über die API Gateway-Konsole einrichten

Eine API-Methode kann über eine oder mehrere Antworten verfügen. Jede Antwort wird von ihrem HTTP-Statuscode indiziert. Standardmäßig fügt die API Gateway-Konsole den Antworten der Methode eine `200`-Antwort hinzu. Sie können dies ändern, z. B. so, dass die Methode stattdessen `201` zurückgibt. Sie können weitere Antworten hinzufügen, z. B. `409` bei Zugriffsverweigerungen und `500` bei Verwendungen nicht initialisierter Stufenvariablen.

Um die API Gateway-Konsole zum Ändern, Löschen oder Hinzufügen einer Antwort zu einer API-Methode zu verwenden, folgen Sie diesen Anweisungen.

1. Wählen Sie im Bereich Ressourcen Ihre Methode und dann die Registerkarte Methodenantwort aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Wählen Sie im Abschnitt Method response settings (Einstellungen für Methodenantwort) die Option Create response (Antwort erstellen) aus.
3. Geben Sie unter HTTP status code (HTTP-Statuscode) einen HTTP-Statuscode wie `200`, `400` oder `500` ein.

   Wenn für eine vom Backend zurückgegebene Antwort keine entsprechende Methodenantwort definiert ist, kann API Gateway die Antwort nicht an den Client zurückgeben. Stattdessen gibt es eine `500 Internal server error`-Fehlerantwort zurück.
4. Wählen Sie Add header.
5. Geben Sie unter Header name (Header-Name) den Namen ein.

   Um einen Header vom Backend an den Client zurückzugeben, fügen Sie den Header in die Methodenantwort ein.
6. Wählen Sie Add model (Modell hinzufügen), um ein Format für den Textkörpers der Methodenantwort zu definieren.

   Geben Sie den Medientyp der Antwortnutzlast für Content type (Inhaltstyp) ein und wählen Sie ein Modell aus dem Dropdown-Menü Models (Modelle) aus.
7. Wählen Sie Speichern.

Um eine bestehende Antwort zu ändern, navigieren Sie zu Ihrer Methodenantwort und wählen Sie dann Edit (Bearbeiten). Um den HTTP status code (HTTP-Statuscode) zu ändern, wählen Sie Delete (Löschen) und erstellen Sie eine neue Methodenantwort.

Für jede vom Backend zurückgegebene Antwort, müssen Sie über eine kompatible Antwort verfügen, die als Methodenantwort konfiguriert ist. Jedoch ist die Konfiguration der Methodenantwort-Header und des Nutzlastmodells optional, es sei denn, Sie weisen das Ergebnis des Backends der Methodenantwort vor der Rückgabe an den Client zu. Zudem ist ein Nutzlastmodell der Methodenantwort wichtig, wenn Sie einen stark typisierten SDK für Ihre API generieren.

Zugriff auf eine REST-API in API Gateway steuern und verwalten

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre API.

Sie können die folgenden Mechanismen für die Authentifizierung und Autorisierung verwenden:

- Mit Ressourcen-Richtlinien können Sie ressourcenbasierte Richtlinien erstellen, um Zugriff auf Ihre APIs und Methoden von angegeben Quell-IP-Adressen oder VPC-Endpunkten aus zu gewähren oder zu verweigern. Weitere Informationen finden Sie unter the section called “API Gateway-Ressourcenrichtlinien verwenden”.
- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen, die auf eine gesamte API oder einzelne Methoden angewendet werden können. IAM-Rollen und -Richtlinien können verwendet werden, um zu steuern, wer Ihre APIs erstellen und verwalten kann und wer sie aufrufen darf. Weitere Informationen finden Sie unter the section called “Verwenden von IAM-Berechtigungen”.
- IAM-Tags können zusammen mit IAM-Richtlinien verwendet werden, um den Zugriff zu steuern. Weitere Informationen finden Sie unter the section called “Attributbasierte Zugriffskontrolle”.
- Endpunktrichtlinien für Schnittstellen-VPC-Endpunkte ermöglichen Ihnen das Anfügen von IAM-Ressourcenrichtlinien an Schnittstellen-VPC-Endpunkte, um die Sicherheit Ihrer privaten APIs zu verbessern. Weitere Informationen finden Sie unter the section called “Verwenden von VPC-Endpunktrichtlinien für private-APIs”.
- Lambda-Genehmiger sind Lambda-Funktionen zur Steuerung des Zugriffs auf Ihre REST-API-Methoden unter Verwendung einer Bearer-Token-Authentifizierung sowie Informationen, die von Headern, Pfaden, Abfragezeichenfolgen, Stufenvariablen oder Kontextvariablen als Anfrageparameter beschrieben werden. Lambda-Genehmiger werden verwendet, um zu steuern, wer REST-API-Methoden aufrufen kann. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden”.
- Amazon Cognito-Benutzerpools ermöglichen Ihnen die Erstellung anpassbarer Authentifizierungs- und Autorisierungslösungen für Ihre REST-APIs. Amazon Cognito-Benutzerpools werden verwendet, um zu steuern, wer REST-API-Methoden aufrufen kann. Weitere Informationen finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”.

Sie können die folgenden Mechanismen verwenden, um andere Aufgaben im Zusammenhang mit der Zugriffssteuerung auszuführen:

- Mittels Cross-Origin Resource Sharing (CORS) können Sie steuern, wie Ihre REST-API auf Cross-Domain-Ressourcenanforderungen reagiert. Weitere Informationen finden Sie unter the section called “CORS”.

- Clientseitige SSL-Zertifikate können verwendet werden, um sicherzustellen, dass HTTP-Anfragen an Ihr Backend-System von API Gateway stammen. Weitere Informationen finden Sie unter the section called “Clientzertifikate”.
- AWS WAF kann verwendet werden, um Ihre API-Gateway-API gegen häufige Web-Exploits zu schützen. Weitere Informationen finden Sie unter the section called “AWS WAF”.

Sie können die folgenden Mechanismen verwenden, um den Zugriff nachzuverfolgen und einzuschränken, den Sie autorisierten Clients gewährt haben:

- Mit Nutzungsplänen können Sie API-Schlüssel für Ihre Kunden bereitstellen und die Verwendung Ihrer API-Stufen und Methoden für jeden API-Schlüssel nachverfolgen und beschränken. Weitere Informationen finden Sie unter the section called “Nutzungspläne”.

Zugriff auf eine API mit API Gateway-Ressourcenrichtlinien steuern

Amazon API Gateway-Ressourcenrichtlinien sind JSON-Richtliniendokumente, die Sie an eine API anhängen, um zu steuern, ob ein bestimmter Prinzipalbenutzer (normalerweise eine IAM-Rolle oder -Gruppe) die API aufrufen kann. Sie können API Gateway-Ressourcenrichtlinien verwenden, damit Ihre API sicher aufgerufen werden kann:

- Benutzer aus einem bestimmten AWS Konto.
- Angegebene Quell-IP-Adressbereiche oder CIDR-Blöcke.
- Angegebene Virtual Private Clouds (VPCs) oder VPC-Endpunkte (in einem beliebigen Konto).

Sie können eine Ressourcenrichtlinie für jeden API-Endpunkttyp in API Gateway anhängen, indem Sie die AWS CLI AWS Management Console, oder AWS SDKs verwenden. Für private APIs können Sie Ressourcenrichtlinien zusammen mit VPC-Endpunktrichtlinien verwenden, um zu steuern, welche Prinzipale Zugriff auf welche Ressourcen und Aktionen erhalten sollen. Weitere Informationen finden Sie unter the section called “Verwenden von VPC-Endpunktrichtlinien für private-APIs”.

API-Gateway-Ressourcenrichtlinien unterscheiden sich von IAM-identitätsbasierten Richtlinien. IAM-identitätsbasierte Richtlinien sind IAM-Benutzern, -Gruppen oder -Rollen zugeordnet und definieren, welche Aktionen diese Identitäten mit welchen Ressourcen ausführen können. API Gateway-Ressourcenrichtlinien werden an Ressourcen angehängt. Sie können API Gateway-Ressourcenrichtlinien zusammen mit IAM-Richtlinien verwenden. Weitere Informationen finden Sie unter Identitätsbasierte und ressourcenbasierte Richtlinien.

Themen

Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway

Diese Seite beschreibt die grundlegenden Elemente, die in den Ressourcenrichtlinien für Amazon API Gateway verwendet werden.

Ressourcenrichtlinien werden mit derselben Syntax wie IAM-Richtlinien spezifiziert. Vollständige Informationen zur Richtliniensprache finden Sie unter Übersicht der IAM-Richtlinien und AWS Identity and Access Management -Richtlinienverweis im IAM-Benutzerhandbuch.

Informationen darüber, wie ein AWS Dienst entscheidet, ob eine bestimmte Anfrage zugelassen oder abgelehnt werden soll, finden Sie unter Feststellen, ob eine Anfrage zugelassen oder verweigert wird.

Allgemeine Elemente einer Zugriffsrichtlinie

In ihrer einfachsten Form enthält eine Ressourcenrichtlinie die folgenden Elemente:

- Ressourcen – APIs sind die Amazon API Gateway-Ressourcen, für die Sie Berechtigungen zulassen oder verweigern können. In einer Richtlinie identifizieren Sie die Ressource mithilfe eines Amazon-Ressourcennamens (ARN). Sie können außerdem eine abgekürzte Syntax verwenden, die API Gateway beim Speichern einer Ressourcenrichtlinie automatisch auf den vollständigen ARN erweitert. Weitere Informationen hierzu finden Sie unter Beispiele für API Gateway-Ressourcenrichtlinien.

  Informationen zum Format des vollständigen `Resource`-Elements finden Sie unter Ressourcenformat für Berechtigungen zur Ausführung der API in API Gateway.
- Aktionen – Für jede Ressource unterstützt Amazon API Gateway eine Reihe von Operationen. Sie identifizieren Ressourcenoperationen, die Sie zulassen (oder ablehnen) können, indem Sie Aktionsschlüsselwörter verwenden.

  Beispielsweise kann der Benutzer mit der `execute-api:Invoke`-Berechtigung die API nach einer Clientanforderung aufrufen.

Informationen zum Format des `Action`-Elements finden Sie unter Aktionsformat für Berechtigungen zur Ausführung der API in API Gateway.

- Auswirkung – Zeigt die Auswirkung, wenn der Benutzer die bestimmten Aktion anfordert – entweder `Allow` oder `Deny`. Sie können den Zugriff auf eine Ressource auch explizit verweigern. Damit können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.

> **Note**
>
> "Implizit verweigern" ist identisch mit "Standardmäßig verweigern".
> Eine "implizite Verweigerung" unterscheidet sich von einer "expliziten Verweigerung".
> Weitere Informationen finden Sie unter Der Unterschied zwischen standardmäßiger und expliziter Zugriffsverweigerung.

- Prinzipal – das Konto oder der Benutzer, das oder der Zugriff auf die Aktionen und Ressourcen in der Anweisung hat. In einer Ressourcenrichtlinie ist der Prinzipal der Benutzer oder das Konto, der bzw. das die Berechtigung erhält.

Die folgende Beispiel-Ressourcenrichtlinie zeigt die zuvor genannten allgemeinen Richtlinienelemente. Die Richtlinie gewährt Zugriff auf alle APIs unter der angegebenen *Konto-ID* in der angegebenen *Region* für jeden Benutzer, dessen Quell-IP-Adresse im Adressblock *123.4.5.6/24* enthalten ist. Die Richtlinie verweigert den Zugriff auf die API, wenn die Quell-IP des Benutzers nicht innerhalb des Bereichs liegt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:*"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:*",
```

```
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "123.4.5.6/24"
                }
            }
        }
    ]
}
```

So beeinflussen API Gateway-Ressourcenrichtlinien den Autorisierungs-Workflow

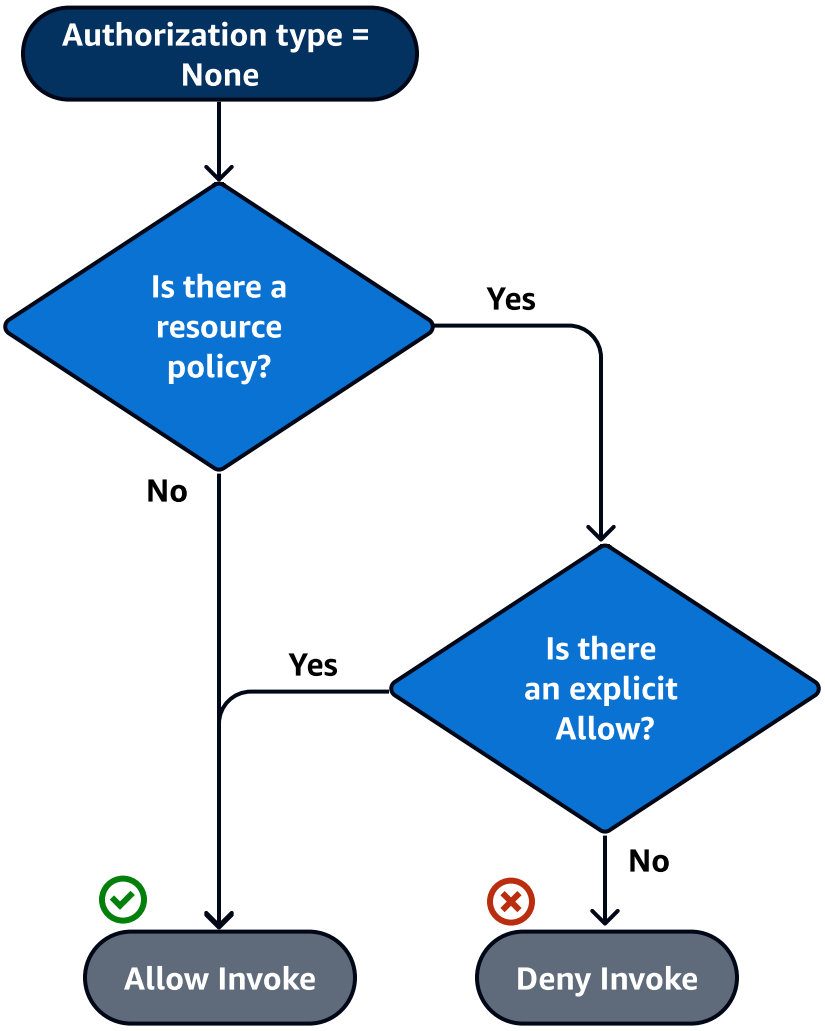
Wenn API Gateway die an die API angefügte Ressourcenrichtlinie ausgewertet wird, wird das Ergebnis vom Authentifizierungstyp beeinflusst, den Sie für die API definiert haben, wie in den Ablaufdiagrammen in den folgenden Abschnitten dargestellt.

Themen

- Nur API Gateway-Ressourcenrichtlinie
- Lambda-Genehmiger und Ressourcenrichtlinie
- IAM-Authentifizierung und Ressourcenrichtlinie
- Amazon Cognito-Authentifizierung und Ressourcenrichtlinie
- Ergebnistabellen für die Richtlinienauswertung

Nur API Gateway-Ressourcenrichtlinie

In diesem Workflow wird eine API Gateway-Ressourcenrichtlinie an die API angehängt. Es wird jedoch kein Authentifizierungstyp für die API definiert. Die Bewertung der Richtlinie beinhaltet das Suchen nach einer expliziten Erlaubnis basierend auf den eingehenden Kriterien des Aufrufers. Eine implizite Verweigerung oder eine explizite Verweigerung führt dazu, dass der Aufrufer abgelehnt wird.



Im Folgenden finden Sie ein Beispiel für eine solche Ressourcenrichtlinie.

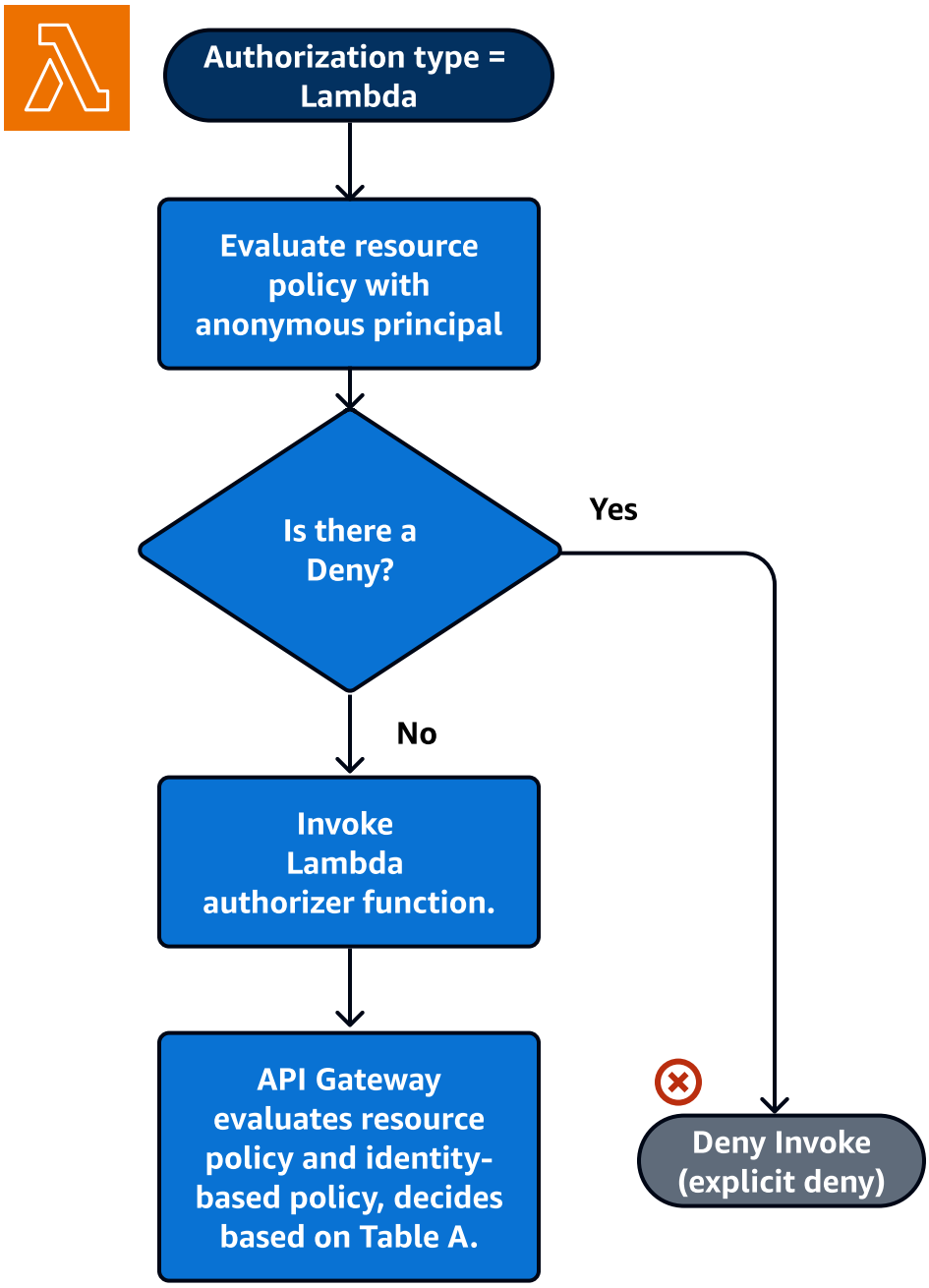
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
```

```
        }
    ]
}
```

Lambda-Genehmiger und Ressourcenrichtlinie

In diesem Workflow wird zusätzlich zu einer Ressourcenrichtlinie ein Lambda-Genehmiger für die API konfiguriert. Die Ressourcenrichtlinie wird in zwei Phasen ausgewertet. Bevor der Lambda-Genehmiger aufgerufen wird, evaluiert API Gateway zunächst die Richtlinie und prüft auf eine explizite Verweigerung. Wenn dies der Fall ist, wird dem Aufrufer der Zugriff sofort verweigert. Andernfalls wird der Lambda-Genehmiger aufgerufen, und es wird ein Richtliniendokument zurückgegeben, das in Verbindung mit der Ressourcenrichtlinie ausgewertet wird. Das Ergebnis wird anhand von Tabelle A ermittelt.

Die folgende Beispielressourcenrichtlinie ermöglicht Aufrufe nur vom VPC-Endpunkt, dessen VPC-Endpunkt-ID lautet*vpce-1a2b3c4d*. Während der "Pre-Auth"-Evaluierung können nur die Aufrufe von dem im Beispiel angegebenen VPC-Endpunkt vorwärts gehen und den Lambda-Genehmiger auswerten. Alle verbleibenden Aufrufe werden blockiert.




```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
```

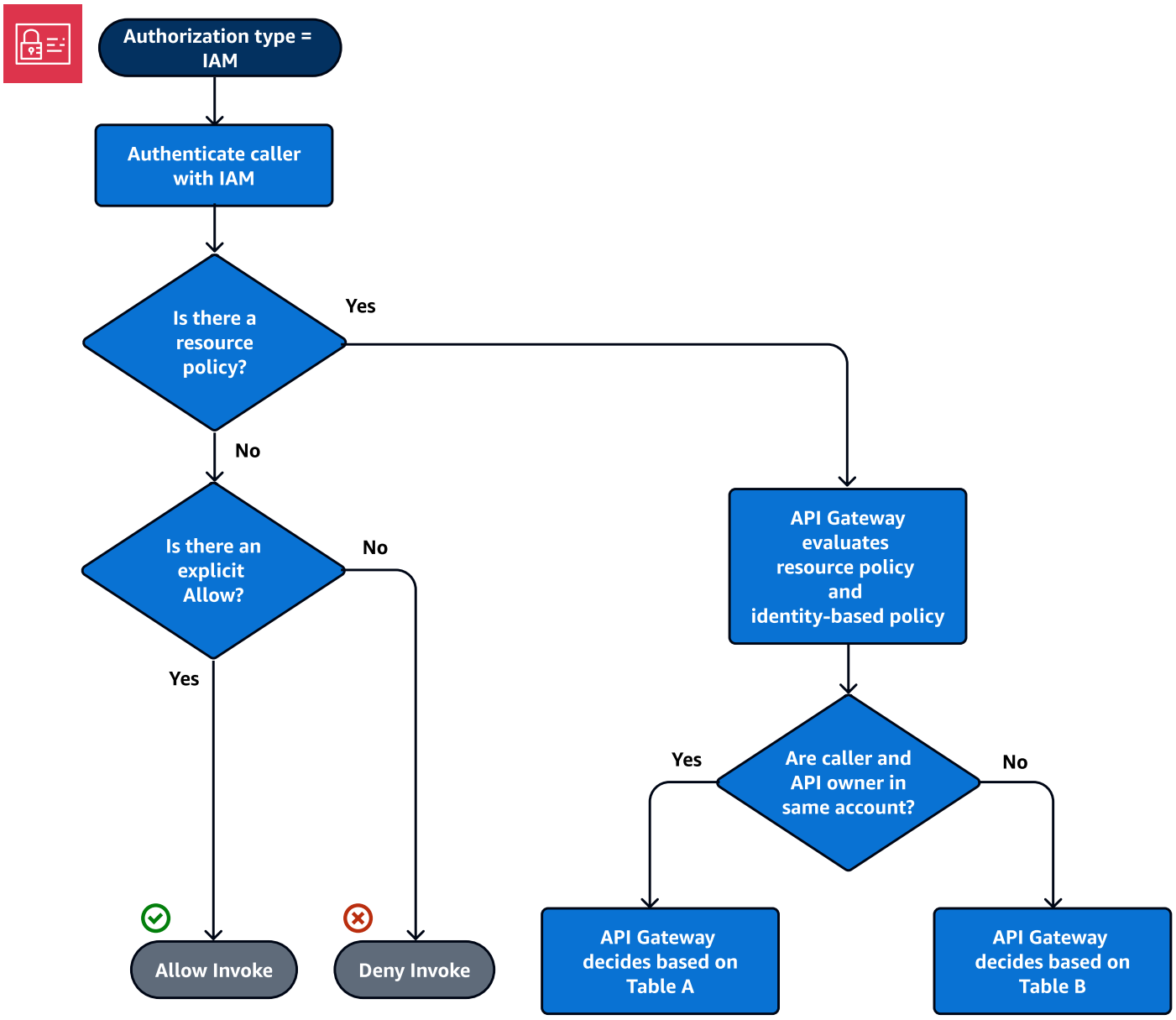
```
                "arn:aws:execute-api:region:account-id:api-id/"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

IAM-Authentifizierung und Ressourcenrichtlinie

In diesem Workflow wird die IAM-Authentifizierung für die API zusätzlich zu einer Ressourcenrichtlinie konfiguriert. Nach dem Authentifizieren des Benutzers mit dem IAM-Service wertet die API die dem Benutzer zugeordneten Richtlinien und die Ressourcenrichtlinie aus. Das Ergebnis hängt davon ab, ob sich der Aufrufer in derselben AWS-Konto oder einer anderen Gruppe AWS-Konto als der API-Besitzer befindet.

Wenn der Aufrufer und der API-Besitzer aus separaten Konten stammen, erlauben sowohl die IAM-Richtlinien als auch die Ressourcenrichtlinie dem Aufrufer explizit, fortzufahren. Weitere Informationen finden Sie in Tabelle B.

Wenn sich der Aufrufer und der API-Besitzer allerdings im selben AWS-Konto befinden, müssen die Benutzerrichtlinien oder die Ressourcenrichtlinie dem Aufrufer ausdrücklich erlauben, fortzufahren. Weitere Informationen finden Sie in Tabelle A.



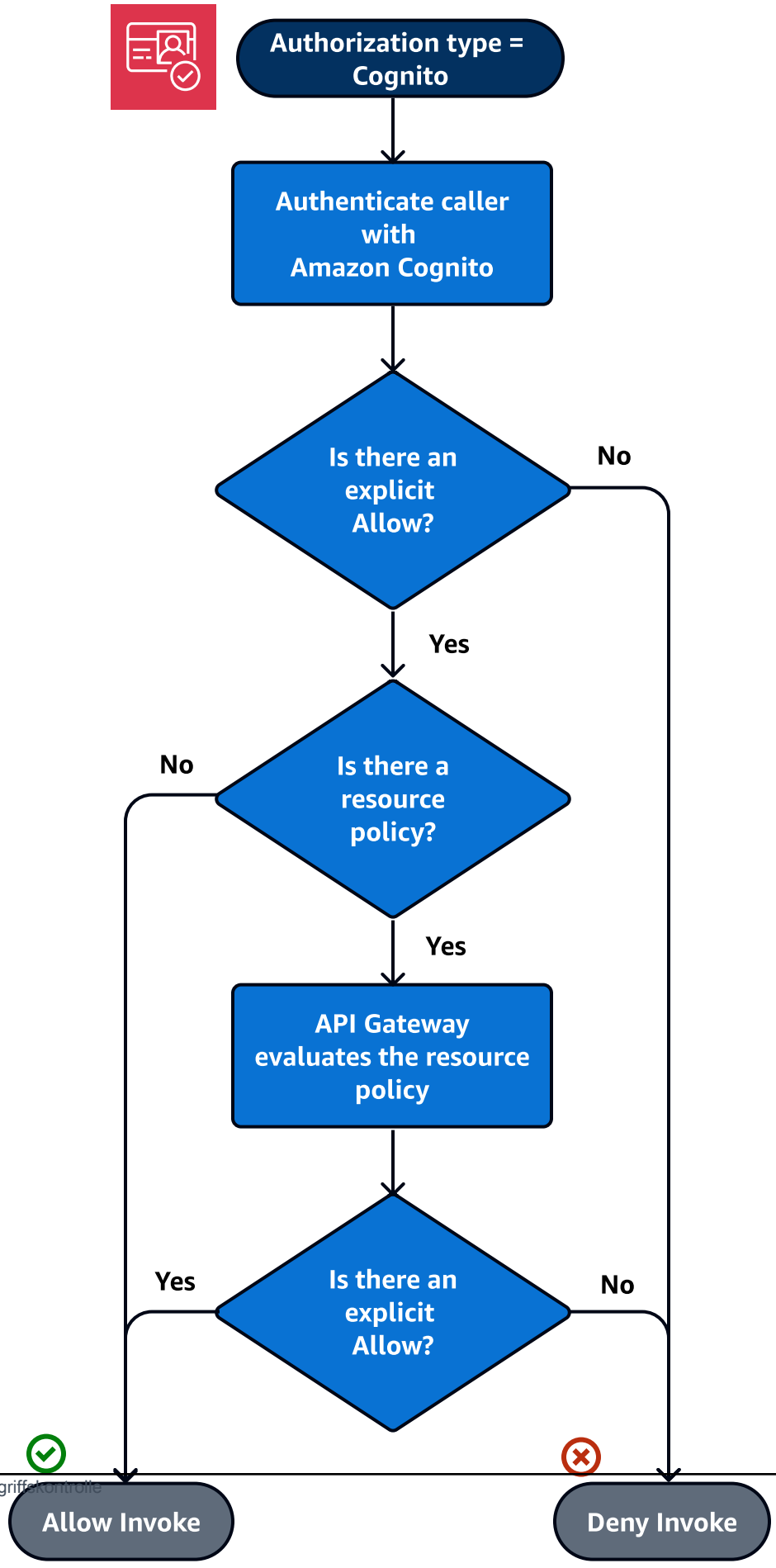
Im Folgenden finden Sie ein Beispiel für eine kontoübergreifende Ressourcenrichtlinie. Unter der Annahme, dass die IAM-Richtlinie einen Zulassungseffekt enthält, erlaubt diese Ressourcenrichtlinie Aufrufe nur von der VPC, deren VPC-ID *vpc-2f09a348* ist. Weitere Informationen finden Sie in Tabelle B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
```

```
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/"
            ],
            "Condition" : {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-2f09a348"
                    }
            }
        }
    ]
}
```

Amazon Cognito-Authentifizierung und Ressourcenrichtlinie

In diesem Workflow wird zusätzlich zu einer Ressourcenrichtlinie ein Amazon Cognito-Benutzerpool für die API konfiguriert. API Gateway versucht zunächst, den Aufrufer über Amazon Cognito zu authentifizieren. Dies wird in der Regel über ein JWT-Token durchgeführt, das vom Aufrufer bereitgestellt wird. Wenn die Authentifizierung erfolgreich ist, wird die Ressourcenrichtlinie unabhängig ausgewertet und eine explizite Genehmigung erforderlich. Eine Zugriffsverweigerung oder weder "Allow" (Zugriffserlaubnis) oder "Deny" (Zugriffsverweigerung) führt zu einer Zugriffsverweigerung. Es folgt ein Beispiel für eine Ressourcenrichtlinie, die zusammen mit Amazon Cognito-Benutzerpools verwendet werden könnte.


Authorization type = Cognito
Authenticate caller with Amazon Cognito
Is there an explicit Allow?
No
Yes
Is there a resource policy?
No
Yes
API Gateway evaluates the resource policy
Is there an explicit Allow?
Yes
No
Allow Invoke
Deny Invoke

Es folgt ein Beispiel für eine Ressourcenrichtlinie, die Aufrufe nur von angegebenen Quell-IPs zulässt, wobei davon ausgegangen wird, dass das Authentifizierungstoken von Amazon Cognito eine Erlaubnis enthält. Weitere Informationen finden Sie in Tabelle B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Ergebnistabellen für die Richtlinienauswertung

In Tabelle A ist das Verhalten aufgeführt, das entsteht, wenn der Zugriff auf eine API-Gateway-API durch eine IAM-Richtlinie oder einen Lambda-Authorizer und eine API-Gateway-Ressourcenrichtlinie gesteuert wird, die sich beide in derselben befinden. AWS-Konto

Tabelle A: Konto A ruft eine API auf, die Konto A gehört

IAM-Richtlinie (oder Lambda-Autorisierer)	API Gateway-Ressourcen richtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

IAM-Richtlinie (oder Lambda-Autorisierer)	API Gateway-Ressourcen richtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Deny	Explizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Weder Zulassen noch Verweigern	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Deny	Explizite Zugriffsverweigerung
Deny	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Explizite Zugriffsverweigerung
Deny	Weder Zulassen noch Verweigern	Explizite Zugriffsverweigerung
Deny	Deny	Explizite Zugriffsverweigerung

In Tabelle B ist das Verhalten aufgeführt, das entsteht, wenn der Zugriff auf eine API-Gateway-API durch eine IAM-Richtlinie gesteuert wird oder ein Amazon Cognito Cognito-Benutzerpool-Autorisierer und eine API Gateway Gateway-Ressourcenrichtlinie, die sich unterscheiden, gesteuert wird. AWS-Konten Wenn einer der beiden Parameter nicht zulässt (weder zulassen noch verweigern), wird der kontoübergreifende Zugriff verweigert. Dies liegt daran, dass der kontoübergreifende Zugriff erfordert, dass sowohl die Ressourcenrichtlinie als auch die IAM-Richtlinie oder der Autorisierer für Amazon Cognito Cognito-Benutzerpools explizit Zugriff gewähren.

Tabelle B: Konto B ruft eine API auf, die Konto A gehört

IAM-Richtlinie (oder Autorisierer für Amazon Cognito Cognito-Benutzerpools)	API Gateway-Ressourcenrichtlinie	Resultierendes Verhalten
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Deny	Explizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Weder Zulassen noch Verweigern	Implizite Zugriffsverweigerung
Weder Zulassen noch Verweigern	Deny	Explizite Zugriffsverweigerung
Deny	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf	Explizite Zugriffsverweigerung
Deny	Weder Zulassen noch Verweigern	Explizite Zugriffsverweigerung
Deny	Deny	Explizite Zugriffsverweigerung

Beispiele für API Gateway-Ressourcenrichtlinien

Auf dieser Seite werden einige Beispiele zu typischen Anwendungsfällen für API Gateway Ressourcenrichtlinien vorgestellt.

In den folgenden Beispielrichtlinien wird eine vereinfachte Syntax verwendet, um die API-Ressource anzugeben. Diese vereinfachte Syntax ist eine abgekürzte Form, mit der Sie auf eine API-Ressource verweisen können, anstatt den vollständigen Amazon-Ressourcenname (ARN) anzugeben. API Gateway konvertiert die abgekürzte Syntax in den vollständigen ARN, wenn Sie die Richtlinie speichern. Sie können zum Beispiel die Ressource `execute-api:/stage-name/GET/pets` in einer Ressourcenrichtlinie angeben. API Gateway wandelt die Ressource in `arn:aws:execute-api:us-east-2:123456789012:aabbccddee/stage-name/GET/pets` um, wenn Sie die Ressourcenrichtlinie speichern. API Gateway erstellt den vollständigen ARN unter Verwendung der aktuellen Region, Ihrer AWS Konto-ID und der ID der REST-API, mit der die Ressourcenrichtlinie verknüpft ist. Sie können mit `execute-api:/*` alle Phasen, Methoden und Pfade in der aktuellen API darstellen. Informationen zur Sprache der Zugriffsichtlinie finden Sie unter Übersicht über die Zugriffsrichtliniensprache für Amazon API Gateway.

Themen

- Beispiel: Erlauben Sie Rollen in einem anderen AWS Konto, eine API zu verwenden
- Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs
- Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs bei Verwendung einer privaten API
- Beispiel: Erlauben von privatem API-Datenverkehr basierend auf der Quell-VPC oder dem VPC-Endpunkt

Beispiel: Erlauben Sie Rollen in einem anderen AWS Konto, eine API zu verwenden

Die folgende Beispiel-Ressourcenrichtlinie gewährt API-Zugriff in einem AWS Konto auf zwei Rollen in einem anderen AWS Konto über Signature Version 4-Protokolle (Sigv4). Insbesondere wird der Entwickler- und der Administratorrolle für das durch *`account-id-2`* identifizierte AWS Konto die `execute-api:Invoke` Aktion zur Ausführung der `GET` Aktion auf der `pets` Ressource (API) in Ihrem AWS Konto gewährt.

```
{
   "Version": "2012-10-17",
```

```
   "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id-2:role/developer",
                    "arn:aws:iam::account-id-2:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs

Die folgende Beispiel-Ressourcenrichtlinie verweigert (blockiert) den eingehenden Datenverkehr zu einer API, wenn dieser von zwei bestimmten Quell-IP-Adressblöcken stammt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
```

```
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs bei Verwendung einer privaten API

Die folgende Beispiel-Ressourcenrichtlinie verweigert (blockiert) den eingehenden Datenverkehr zu einer privaten API, wenn dieser von zwei bestimmten Quell-IP-Adressblöcken stammt. Bei Verwendung privater APIs schreibt der VPC-Endpunkt für `execute-api` die ursprüngliche Quell-IP-Adresse neu. Die `aws:VpcSourceIp`-Bedingung filtert die Anforderung anhand der ursprünglichen IP-Adresse des Anforderers.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

Beispiel: Erlauben von privatem API-Datenverkehr basierend auf der Quell-VPC oder dem VPC-Endpunkt

Die folgenden Beispiel-Ressourcenrichtlinien erlauben eingehenden Datenverkehr zu einer privaten API von nur einer bestimmten Virtual Private Cloud (VPC) oder einem bestimmten VPC-Endpunkt.

In diesem Beispiel für eine Ressourcenrichtlinie wird eine Quell-VPC angegeben:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-1a2b3c4d"
                }
            }
        }
    ]
}
```

In diesem Beispiel für eine Ressourcenrichtlinie wird ein Quell-VPC-Endpunkt angegeben:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
```

```
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

API Gateway-Ressourcenrichtlinie erstellen und an eine API anhängen

Um einem Benutzer den Zugriff auf Ihre API zu ermöglichen, indem er den API-Ausführungsdienst aufruft, müssen Sie eine API-Gateway-Ressourcenrichtlinie erstellen und die Richtlinie an die API anhängen. Wenn Sie eine Richtlinie an Ihre API anhängen, werden die in der Richtlinie enthaltenen Berechtigungen auf die Methoden in der API angewendet. Wenn Sie die Ressourcenrichtlinie aktualisieren, müssen Sie die API bereitstellen.

Themen

- Voraussetzungen
- Eine Ressourcenrichtlinie an eine API Gateway anhängen
- Beheben Sie Fehler bei Ihrer Ressourcenrichtlinie

Voraussetzungen

Um eine API Gateway Gateway-Ressourcenrichtlinie zu aktualisieren, benötigen Sie die `apigateway:UpdateRestApiPolicy` Genehmigung und die `apigateway:PATCH` Erlaubnis.

Bei einer Edge-optimierten oder regionalen API können Sie Ihre Ressourcenrichtlinie an Ihre API anhängen, während Sie sie erstellen oder nachdem sie bereitgestellt wurde. Bei einer privaten API

können Sie Ihre API nicht ohne eine Ressourcenrichtlinie bereitstellen. Weitere Informationen finden Sie unter the section called “Private REST-APIs”.

Eine Ressourcenrichtlinie an eine API Gateway anhängen

Das folgende Verfahren zeigt Ihnen, wie Sie eine Ressourcenrichtlinie an eine API Gateway anhängen.

AWS Management Console

So hängen Sie eine Ressourcenrichtlinie an eine API Gateway-API an:

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Wählen Sie Richtlinie erstellen aus.
5. (Optional) Wählen Sie Vorlage auswählen, um eine Beispielrichtlinie zu generieren.

   In den Beispielrichtlinien sind Platzhalter in doppelte geschweifte Klammern ("{{*placeholder*}}") eingeschlossen. Ersetzen Sie die einzelnen Platzhalter (einschließlich der geschweiften Klammern) durch die erforderlichen Informationen.
6. Wenn Sie keines der Vorlagenbeispiele verwenden, geben Sie Ihre Ressourcenrichtlinie ein.
7. Wählen Sie Änderungen speichern aus.

Wenn die API zuvor in der API Gateway-Konsole bereitgestellt wurde, müssen Sie sie neu bereitstellen, damit die Ressourcenrichtlinie wirksam wird.

AWS CLI

Rufen Sie den create-rest-apiBefehl wie folgt AWS CLI auf, um eine neue API zu erstellen und ihr eine Ressourcenrichtlinie anzuhängen:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy "{\"jsonEscapedPolicyDocument\"}"
```

Rufen Sie den AWS CLI update-rest-apiBefehl wie folgt auf, um eine Ressourcenrichtlinie an eine bestehende API anzuhängen:

```
aws apigateway update-rest-api \
    --rest-api-id api-id \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

AWS CloudFormation

Sie können verwenden AWS CloudFormation , um eine API mit einer Ressourcenrichtlinie zu erstellen. Das folgende Beispiel erstellt eine REST-API mit der Beispiel-Ressourcenrichtlinie,the section called “Beispiel: Verweigern des API-Datenverkehrs basierend auf der Quell-IP-Adresse oder eines IP-Bereichs”.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api/*'
            Condition:
              IpAddress:
                'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
        Version: 2012-10-17
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
```

```
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: MOCK
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - MethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: test
```

Beheben Sie Fehler bei Ihrer Ressourcenrichtlinie

Die folgenden Anleitungen zur Fehlerbehebung können Ihnen helfen, Probleme mit Ihrer Ressourcenrichtlinie zu lösen.

Meine API gibt {"Message“ zurück: "User: anonymous is not authorized to perform: execute-api:invoke on resource: arn:aws:execute-api:us-east-1: ********/****/****/ "}

Wenn Sie in Ihrer Ressourcenrichtlinie den AWS Principal auf einen Principal setzen, wie zum Beispiel den folgenden:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            ""Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:role/developer",
                    "arn:aws:iam::account-id:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        ...
```

```
        }
```

Sie müssen die `AWS_IAM` Autorisierung für jede Methode in Ihrer API verwenden, andernfalls gibt Ihre API die vorherige Fehlermeldung zurück. Weitere Anweisungen zum Aktivieren der `AWS_IAM` Autorisierung für eine Methode finden Sie unterthe section called “Methoden”.

Meine Ressourcenrichtlinie wird nicht aktualisiert

Wenn Sie die Ressourcenrichtlinie nach dem Erstellen der API aktualisieren, müssen Sie die API bereitstellen, damit die Änderungen übernommen werden, nachdem Sie die aktualisierte Richtlinie angefügt haben. Das Aktualisieren oder Speichern der Richtlinie allein ändert das Laufzeitverhalten der API nicht. Weitere Informationen zum Bereitstellen Ihrer API finden Sie unter the section called “Bereitstellen einer REST-API”.

AWS Bedingungsschlüssel, die in API-Gateway-Ressourcenrichtlinien verwendet werden können

Die folgende Tabelle enthält AWS Bedingungsschlüssel, die in Ressourcenrichtlinien für APIs in API Gateway für jeden Autorisierungstyp verwendet werden können.

Weitere Informationen zu AWS Bedingungsschlüsseln finden Sie unter AWS Globale Bedingungskontextschlüssel.

Tabelle der Bedingungsschlüssel

Bedingungsschlüssel	Kriterien	Erfordert **`AuthN`**?	Autorisierungstyp
`aws:CurrentTime`	Keine	Nein	Alle
`aws:EpochTime`	Keine	Nein	Alle
`aws:TokenIssueTime`	Schlüssel ist nur in Anforderungen vorhanden, die mithilfe temporärer Sicherheitsanmeldeinformationen signiert werden.	Ja	IAM
`aws:MultiFactorAuthPresent`	Schlüssel ist nur in Anforderungen vorhanden, die	Ja	IAM

Bedingungsschlüssel	Kriterien	Erfordert **AuthN**?	Autorisierungstyp
	mithilfe temporärer Sicherheitsanmelde informationen signiert werden.		
`aws:MultiFactorAuthAge`	Schlüssel ist nur vorhanden, wenn die MFA in den Anforderungen enthalten ist.	Ja	IAM
`aws:PrincipalAccount`	Keine	Ja	IAM
`aws:PrincipalArn`	Keine	Ja	IAM
`aws:PrincipalOrgID`	Dieser Schlüssel ist nur dann im Anforderungskontext enthalten, wenn der Prinzipal Mitglied einer Organisation ist.	Ja	IAM
`aws:PrincipalOrgPaths`	Dieser Schlüssel ist nur dann im Anforderungskontext enthalten, wenn der Prinzipal Mitglied einer Organisation ist.	Ja	IAM

Bedingungsschlüssel	Kriterien	Erfordert **AuthN**?	Autorisierungstyp
`aws:PrincipalTag`	Dieser Schlüssel ist nur dann im Anforderungskontext enthalten, wenn der Prinzipal ein IAM-Benutzer mit angefügten Tags ist. Er ist für einen Prinzipal enthalten, der eine IAM-Rolle mit angefügten Tags oder Sitzungs-Tags verwendet.	Ja	IAM
`aws:PrincipalType`	Keine	Ja	IAM
`aws:Referer`	Schlüssel ist nur vorhanden, wenn der Wert vom Aufrufer im HTTP-Header bereitgestellt wird.	Nein	Alle
`aws:SecureTransport`	Keine	Nein	Alle
`aws:SourceArn`	Keine	Nein	Alle
`aws:SourceIp`	Keine	Nein	Alle
`aws:SourceVpc`	Dieser Schlüssel kann nur für private APIs verwendet werden.	Nein	Alle
`aws:SourceVpce`	Dieser Schlüssel kann nur für private APIs verwendet werden.	Nein	Alle

Bedingungsschlüssel	Kriterien	Erfordert **AuthN**?	Autorisierungstyp
`aws:VpcSourceIp`	Dieser Schlüssel kann nur für private APIs verwendet werden.	Nein	Alle
`aws:UserAgent`	Schlüssel ist nur vorhanden, wenn der Wert vom Aufrufer im HTTP-Header bereitgestellt wird.	Nein	Alle
`aws:userid`	Keine	Ja	IAM
`aws:username`	Keine	Ja	IAM

Kontrollieren des Zugriffs auf eine API mit IAM-Berechtigungen

Sie steuern den Zugriff auf Ihre Amazon API Gateway-API über IAM-Berechtigungen, indem Sie den Zugriff auf die folgenden zwei API Gateway-Komponentenprozesse steuern:

- Um eine API in API Gateway zu erstellen, bereitzustellen und zu verwalten, müssen Sie dem API-Entwickler Berechtigungen zur Durchführung der erforderlichen Aktionen erteilen, die von der API-Verwaltungskomponente von API Gateway unterstützt werden.
- Zum Aufrufen einer bereitgestellten API oder zum Aktualisieren der API-Zwischenspeicherung benötigt der API-Aufrufer die Berechtigungen für die erforderlichen IAM-Aktionen, die von der API-Ausführungskomponente von Amazon API Gateway unterstützt werden.

Die Zugriffskontrolle für die beiden Prozesse umfasst verschiedene Berechtigungsmodelle, die nachstehend erläutert werden.

API Gateway-Berechtigungsmodell für die Erstellung und Verwaltung einer API

Wenn Sie einem API-Entwickler ermöglichen möchten, eine API in Amazon API Gateway zu erstellen und zu verwalten, müssen Sie IAM-Berechtigungsrichtlinien erstellen, die zulassen, dass ein bestimmter API-Entwickler die erforderlichen API-Entitäten erstellen, aktualisieren, bereitstellen, ansehen oder löschen darf. Sie fügen die Richtlinie an einen Benutzer, eine Rolle oder eine Gruppe an.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Weitere Informationen dazu, wie Sie dieses Berechtigungsmodell verwenden, finden Sie unter the section called “Identitätsbasierte API-Gateway-Richtlinien”.

API Gateway-Berechtigungsmodell für den Aufruf einer API

Wenn Sie zulassen möchten, dass ein API-Aufrufer die API aufruft oder deren Cache aktualisiert, müssen Sie IAM-Richtlinien erstellen, die einen bestimmten API-Aufrufer die API-Methode aufrufen lassen, für die die Benutzerauthentifizierung aktiviert ist. Der API-Entwickler legt `authorizationType` als `AWS_IAM`-Eigenschaft der Methode fest, damit der Aufrufer gezwungen ist, für die Authentifizierung die Anmeldeinformationen des Benutzers zu übermitteln. Anschließend verknüpfen Sie die Richtlinie mit einem Benutzer, einer Rolle oder einer Gruppe.

In dieser IAM-Berechtigungsrichtlinienanweisung enthält das IAM-`Resource`-Element eine Liste mit bereitgestellten API-Methoden, die durch bestimmte HTTP-Verben und API Gateway-Ressourcenpfade identifiziert werden. Das IAM `Action`-Element enthält die erforderliche API Gateway-API, die Aktionen ausführt. Zu diesen Aktionen gehören `execute-api:Invoke` oder `execute-api:InvalidateCache`, wobei `execute-api` die zugrundeliegende API-Ausführungskomponente von API Gateway bezeichnet.

Weitere Informationen dazu, wie Sie dieses Berechtigungsmodell verwenden, finden Sie unter Kontrollieren des Zugriffs für den API-Aufruf.

Wenn eine API in einen AWS Dienst (z. B. AWS Lambda) im Backend integriert ist, muss API Gateway auch über Berechtigungen für den Zugriff auf integrierte AWS Ressourcen (z. B. das Aufrufen einer Lambda-Funktion) im Namen des API-Aufrufers verfügen. Um diese Berechtigungen zu erteilen, erstellen Sie eine IAM-Rolle vom Typ AWS -Service für API Gateway. Wenn Sie diese Rolle in der IAM-Managementkonsole erstellen, enthält sie die folgende IAM-Vertrauensrichtlinie. In dieser wird Amazon API Gateway als vertrauenswürdige Entity deklariert, der die Übernahme der Rolle gestattet ist:

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

Wenn Sie die IAM-Rolle durch Aufruf des Befehls create-role der CLI oder einer entsprechenden SDK-Methode erstellen, müssen Sie die oben beschriebene Vertrauensrichtlinie als Eingabeparameter von `assume-role-policy-document` bereitstellen. Versuchen Sie nicht, eine solche Richtlinie direkt in der IAM-Managementkonsole zu erstellen oder den Befehl AWS CLI create-policy oder eine entsprechende SDK-Methode aufzurufen.

Damit API Gateway den integrierten AWS Dienst aufrufen kann, müssen Sie dieser Rolle auch entsprechende IAM-Berechtigungsrichtlinien für den Aufruf integrierter AWS Dienste zuordnen. Um beispielsweise eine Lambda-Funktion aufrufen zu können, müssen Sie die folgende IAM-Berechtigungsrichtlinie in die IAM-Rolle einfügen:

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

```
        }
    ]
}
```

Lambda unterstützt ressourcenbasierte Zugriffsrichtlinien und kombiniert somit Vertrauens- und Berechtigungsrichtlinien. Bei der Integration einer API mit einer Lambda-Funktion über die API Gateway-Konsole werden Sie nicht aufgefordert, diese IAM-Rolle explizit festzulegen, da die Konsole die ressourcenbasierten Berechtigungen für die Lambda-Funktion mit Ihrer Zustimmung für Sie festlegt.

> **Note**
>
> Um die Zugriffskontrolle für einen AWS Dienst festzulegen, können Sie entweder das aufruferbasierte Berechtigungsmodell verwenden, bei dem eine Berechtigungsrichtlinie direkt an den Benutzer oder die Gruppe des Anrufers angehängt wird, oder das rollenbasierte Berechtigungsmodell, bei dem eine Berechtigungsrichtlinie an eine IAM-Rolle angehängt wird, die API Gateway übernehmen kann. Die Berechtigungsrichtlinien in den zwei Modellen können sich unterscheiden. Zum Beispiel kann die aufruferbasierte Richtlinie den Zugriff blockieren, während die rollenbasierte ihn zulässt. Sie können dies nutzen, um zu verlangen, dass ein Benutzer nur über eine API Gateway auf einen AWS Dienst zugreift.

Kontrollieren des Zugriffs für den API-Aufruf

In diesem Abschnitt erfahren Sie, wie Sie IAM-Richtlinienanweisungen verfassen, um zu steuern, wer eine bereitgestellte API in API Gateway aufrufen kann. Sie finden hier auch die Richtlinienanweisungsreferenz, einschließlich der Formate für die `Action`- und `Resource`-Felder für den API-Ausführungsservice. Sie sollten außerdem den IAM-Abschnitt in the section called “Wie Ressourcenrichtlinien den Autorisierungsworkflow beeinflussen” lesen.

Für private APIs sollten Sie eine Kombination aus einer API Gateway-Ressourcenrichtlinie und einer VPC-Endpunktrichtlinie verwenden. Weitere Informationen finden Sie unter den folgenden Themen:

- the section called “API Gateway-Ressourcenrichtlinien verwenden”
- the section called “Verwenden von VPC-Endpunktrichtlinien für private-APIs”

Mit IAM-Richtlinien steuern, wer eine API Gateway-API-Methode aufrufen kann

Um mit IAM-Berechtigungen zu steuern, wer eine bereitgestellte API aufrufen kann oder nicht, erstellen Sie ein IAM-Richtliniendokument mit den erforderlichen Berechtigungen. Eine Vorlage für ein solches Richtliniendokument sehen Sie im Folgenden.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Permission",
      "Action": [
        "execute-api:Execution-operation"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage/METHOD_HTTP_VERB/Resource-path"
      ]
    }
  ]
}
```

Hier muss *`Permission`* durch `Allow` oder `Deny` ersetzt werden, abhängig davon, ob Sie die enthaltenen Berechtigungen erteilen oder entziehen möchten.*`Execution-operation`* ist durch die vom API-Ausführungsdienst unterstützten Operationen zu ersetzen.*`METHOD_HTTP_VERB`* steht für ein HTTP-Verb, das von den angegebenen Ressourcen unterstützt wird. *`Resource-path`* ist der Platzhalter für den URL-Pfad einer bereitgestellten API-`Resource`-Instance, die das genannte *`METHOD_HTTP_VERB`* unterstützt. Weitere Informationen finden Sie unter Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway.

> **Note**
>
> Damit IAM-Richtlinien wirksam sind, müssen Sie die IAM-Authentifizierung bei API-Methoden aktiviert haben, indem Sie `AWS_IAM` für die Eigenschaft `authorizationType` der Methoden festlegen. Andernfalls werden diese API-Methoden öffentlich zugänglich gemacht.

Angenommen, Sie möchten einem Benutzer die Berechtigung erteilen, eine Liste mit Haustieren anzusehen, die von einer bestimmten API zur Verfügung gestellt wird. Der Benutzer darf der Liste

aber kein Haustier hinzufügen. Für diesen Fall können Sie die folgende Anweisung in die IAM-Richtlinie einschließen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/POST/pets"
      ]
    }
  ]
}
```

Um einem Benutzer die Berechtigung zum Anzeigen eines bestimmten Haustiers zu erteilen, das von einer API zur Verfügung gestellt wird, die als `GET /pets/{petId}` konfiguriert ist, können Sie die folgende Anweisung in die IAM-Richtlinie einschließen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets/a1b2"
      ]
```

```
    }
  ]
}
```

Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway

Im Folgenden werden die "Action"- und "Resource"-Formate von IAM-Richtlinienanweisungen für Zugriffsberechtigungen zur Ausführung einer API beschrieben.

Aktionsformat für Berechtigungen zur Ausführung der API in API Gateway

Der `Action`-Ausdruck zur API-Ausführung weist das folgende allgemeine Format auf:

```
execute-api:action
```

Wobei *`action`* eine verfügbare API-Ausführungsaktion ist:

- *, repräsentiert alle der folgenden Aktionen.
- Invoke – wird verwendet, um eine API für eine Client-Anforderung aufzurufen.
- InvalidateCache, wird verwendet, um den API-Cache auf Anfrage eines Clients für ungültig zu erklären.

Ressourcenformat für Berechtigungen zur Ausführung der API in API Gateway

Der `Resource`-Ausdruck zur API-Ausführung weist das folgende allgemeine Format auf:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-
specifier
```

Wobei:

- *`Region`* ist die AWS Region (z. B. **`us-east-1`** oder **`*`** für alle AWS Regionen), die der bereitgestellten API für die Methode entspricht.
- *`Account-ID`* ist die 12-stellige AWS Konto-ID des REST-API-Besitzers.
- *`api-id`* ist die ID, die API Gateway der API für die Methode zugewiesen hat.
- *`stage-name`* ist der Name der Stufe, die mit der Methode verknüpft ist.
- *`HTTP-VERB`* das HTTP-Verb für die Methode ist. Es kann eines der folgenden Verben sein: GET, POST, PUT, DELETE, PATCH.

- *resource-path-specifier*ist der Pfad zur gewünschten Methode.

> **Note**
>
> Wenn Sie einen Platzhalter (*) angeben, wendet der `Resource`-Ausdruck den Platzhalter auf den Rest des Ausdrucks an.

Hier einige Ressourcenausdrücke als Beispiel:

- **`arn:aws:execute-api:*:*:*`**für jeden Ressourcenpfad in jeder Phase, für jede API in jeder AWS Region.
- **`arn:aws:execute-api:us-east-1:*:*`**für jeden Ressourcenpfad in jeder Phase, für jede API in der AWS Region von`us-east-1`.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/*`**für jeden Ressourcenpfad in jeder Phase, für die API mit der ID *`api-id`* in der AWS Region us-east-1.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/test/*`** für einen Ressourcenpfad in der Stufe `test`, für die API mit dem Bezeichner *`api-id`* in der AWS -Region „us-east-1“.

Weitere Informationen hierzu finden Sie unter Referenz zu API Gateway Amazon-Ressourcenname (ARN).

IAM-Richtlinienbeispiele für API-Ausführungsberechtigungen

Weitere Informationen zu Berechtigungsmodellen und mehr Hintergrundwissen erhalten Sie unter Kontrollieren des Zugriffs für den API-Aufruf.

Mit der folgenden Richtlinienanweisung erhält der Benutzer die Berechtigung zum Aufrufen jeder POST-Methode entlang dem Pfad von `mydemoresource` in der `test`-Stufe für die API mit der ID `a123456789`. Dabei wird vorausgesetzt, dass die entsprechende API in der AWS -Region us-east-1 bereitgestellt wurde:

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

```
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/mydemoresource/*"
      ]
    }
  ]
}
```

Die folgende Beispielrichtlinienanweisung erteilt dem Benutzer die Berechtigung zum Aufrufen jeder Methode im Ressourcenpfad von `petstorewalkthrough/pets` in allen Stufen, für die API mit der ID `a123456789`, in jeder AWS -Region, in der die entsprechende API bereitgestellt wurde:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Erstellen einer Richtlinie und Anfügen der Richtlinie an einen Benutzer

Wenn Sie einem Benutzer den Aufruf des API-Verwaltungsservice oder API-Ausführungsservice ermöglichen möchten, müssen Sie eine IAM-Richtlinie erstellen, die den Zugriff auf die API-Gateway-Entitäten kontrolliert.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).

Wenn Sie zum ersten Mal Policies (Richtlinien) auswählen, erscheint die Seite Welcome to Managed Policies (Willkommen bei verwalteten Richtlinien). Wählen Sie Get Started.

3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben Sie folgendes JSON-Richtliniendokument ein:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    }
  ]
}
```

6. Wählen Sie Weiter aus.

> **Note**
>
> Sie können jederzeit zwischen den Editoroptionen Visual und JSON wechseln. Wenn Sie jedoch Änderungen vornehmen oder im Visual-Editor Weiter wählen, strukturiert IAM Ihre Richtlinie möglicherweise um, um sie für den visuellen Editor zu optimieren. Weitere Informationen finden Sie unter Richtlinienrestrukturierung im IAM-Benutzerhandbuch.

7. Geben Sie auf der Seite Prüfen und erstellen unter Richtlinienname einen Namen und unter Beschreibung (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
8. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.

Ersetzen Sie in dieser Anweisung *`action-statement`* und *`resource-statement`* nach Bedarf. Fügen Sie weitere Anweisungen hinzu, um die Amazon API Gateway-Entitäten anzugeben, die der Benutzer verwalten darf, die Methoden, die er aufrufen darf, oder beides. Standardmäßig hat der Benutzer keine Berechtigungen, es sei denn, es gibt eine entsprechende explizite `Allow`-Anweisung.

Sie haben soeben eine IAM-Richtlinie erstellt. Sie hat keine Wirkung, bis Sie sie anfügen.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Fügen Sie ein IAM-Richtliniendokument wie folgt an eine IAM-Gruppe an:

1. Wählen Sie im Haupt-Navigationsbereich Groups aus.
2. Wählen Sie die Registerkarte Permissions unter der gewählten Gruppe aus.
3. Wählen Sie Attach policy aus.

4. Wählen Sie das zuvor von Ihnen erstellte Richtliniendokument aus und klicken Sie auf Attach policy.

Damit API Gateway in Ihrem Namen andere AWS Dienste aufrufen kann, erstellen Sie eine IAM-Rolle vom Typ Amazon API Gateway.

So erstellen Sie eine Rolle vom Typ „Amazon API Gateway“:

1. Wählen Sie im Haupt-Navigationsbereich Roles aus.
2. Klicken Sie auf Create New Role.
3. Geben Sie einen Namen für Role name ein und wählen Sie dann Next Step aus.
4. Wählen Sie unter Rollentyp auswählen, in AWS -Service-Rollen neben Amazon API Gateway die Option Select aus.
5. Wählen Sie eine verfügbare Richtlinie für verwaltete IAM-Berechtigungen aus, z. B. AmazonAPI, GatewayPushToCloudWatchLog wenn Sie möchten, dass API Gateway Metriken protokolliert CloudWatch, unter Richtlinie anhängen und wählen Sie dann Next Step aus.
6. Stellen Sie unter Trusted Entities sicher, dass apigateway.amazonaws.com als Eintrag aufgelistet wird, und wählen Sie dann Create Role aus.
7. Wählen Sie in der neu erstellten Rolle die Registerkarte Permissions und klicken Sie auf Attach Policy.
8. Wählen Sie das zuvor von Ihnen erstellte IAM-Richtliniendokument aus und klicken Sie auf Attach Policy (Richtlinie anhängen).

VPC-Endpunktrichtlinien für private APIs in API Gateway verwenden

Um die Sicherheit Ihrer privaten API zu verbessern, können Sie eine VPC-Endpunktrichtlinie erstellen. Eine VPC-Endpunktrichtlinie ist eine IAM-Ressourcenrichtlinie, die Sie Ihrem VPC-Endpunkt anfügen können. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit VPC-Endpunkten.

Möglicherweise möchten Sie eine VPC-Endpunktrichtlinie erstellen, um Folgendes zu tun:

- Erlauben Sie nur bestimmten Organisationen oder Ressourcen, auf Ihren VPC-Endpunkt zuzugreifen und Ihre API aufzurufen.
- Verwenden Sie eine einzige Richtlinie und vermeiden Sie sitzungs- oder rollenbasierte Richtlinien, um den Datenverkehr zu Ihrer API zu kontrollieren.

- Verschärfen Sie den Sicherheitsbereich Ihrer Anwendung bei der Migration von lokal zu. AWS

Erwägungen zur VPC-Endpunktrichtlinie

- Die Identität des Aufrufers wird anhand des `Authorization`-Header-Werts bewertet. Je nach Ihrem `authorizationType` kann dies zu einem `403 IncompleteSignatureException` oder einem `403 InvalidSignatureException`-Fehler führen. Die folgende Tabelle zeigt die `Authorization`-Header-Werte für die einzelnen `authorizationType`.

`authorizationType`	**`Authorization`** -Header ausgewertet?	Zulässige **`Authorization`** -Header-Werte
`NONE` mit der Standardr ichtlinie für Vollzugriff	Nein	Nicht bestanden
`NONE` mit einer benutzerd efinierten Zugriffsrichtlinie	Ja	Muss ein gültiger SigV4-Wert sein
`IAM`	Ja	Muss ein gültiger SigV4-Wert sein
`CUSTOM` oder `COGNITO_U SER_POOLS`	Nein	Nicht bestanden

- Wenn eine Richtlinie den Zugriff auf einen bestimmten IAM-Prinzipal einschränkt, müssen Sie beispielsweise `arn:aws:iam::account-id:role/developer` die Methode `authorizationType` Ihrer API auf oder setzen. `AWS_IAM` `NONE` Weitere Anweisungen zum Einstellen der `authorizationType` für eine Methode finden Sie unter. the section called “Methoden”
- VPC-Endpunktrichtlinien können zusammen mit API Gateway-Ressourcenrichtlinien verwendet werden. Die API-Gateway-Ressourcenrichtlinie legt fest, welche Principals auf die API zugreifen können. Die Endpunktrichtlinie legt fest, wer auf die VPC zugreifen kann und welche APIs vom VPC-Endpunkt aus aufgerufen werden können. Ihre private API benötigt eine Ressourcenrichtlinie, aber Sie müssen keine benutzerdefinierte VPC-Endpunktrichtlinie erstellen.

Beispiele für VPC-Endpunktrichtlinien

Sie können Richtlinien für Amazon Virtual Private Cloud-Endpunkte für Amazon API Gateway erstellen, in denen Sie folgendes angeben können:

- Prinzipal, der die Aktionen ausführen kann
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Um die Richtlinie dem VPC-Endpunkt anzufügen, müssen Sie die VPC-Konsole verwenden. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit VPC-Endpunkten.

Beispiel 1: VPC-Endpunktrichtlinie, die Zugriff auf zwei APIs gewährt

Die folgende Beispielrichtlinie gewährt über den VPC-Endpunkt, dem die Richtlinie angefügt ist, nur Zugriff auf zwei bestimmte APIs.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*",
                "arn:aws:execute-api:us-east-1:123412341234:aaaaa11111/*"
            ]
        }
    ]
}
```

Beispiel 2: VPC-Endpunktrichtlinie, die Zugriff auf GET-Methoden gewährt

Die folgende Beispielrichtlinie gewährt Benutzern über den VPC-Endpunkt, dem die Richtlinie angefügt ist, Zugriff auf `GET`-Methoden für eine bestimmte API.

```
{
    "Statement": [
```

```
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/GET/*"
            ]
        }
    ]
}
```

Beispiel 3: VPC-Endpunktrichtlinie, die einem bestimmten Benutzer Zugriff auf eine bestimmte API gewährt

Die folgende Beispielrichtlinie gewährt über den VPC-Endpunkt, dem die Richtlinie angefügt ist, einem bestimmten Benutzer Zugriff auf eine bestimmte API.

In diesem Fall müssen Sie die Methode auf oder setzen, da die Richtlinie den Zugriff auf bestimmte IAM-Prinzipale einschränkt. `authorizationType` `AWS_IAM` `NONE`

```
{
    "Statement": [
        {
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123412341234:user/MyUser"
                ]
            },
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*"
            ]
        }
    ]
}
```

Tags zur Steuerung des Zugriffs auf eine REST-API in API Gateway verwenden

Die Berechtigung für den Zugriff auf REST-APIs kann mit der attributbasierten Zugriffskontrolle in IAM-Richtlinien verfeinert werden.

Weitere Informationen finden Sie unter the section called “Attributbasierte Zugriffskontrolle”.

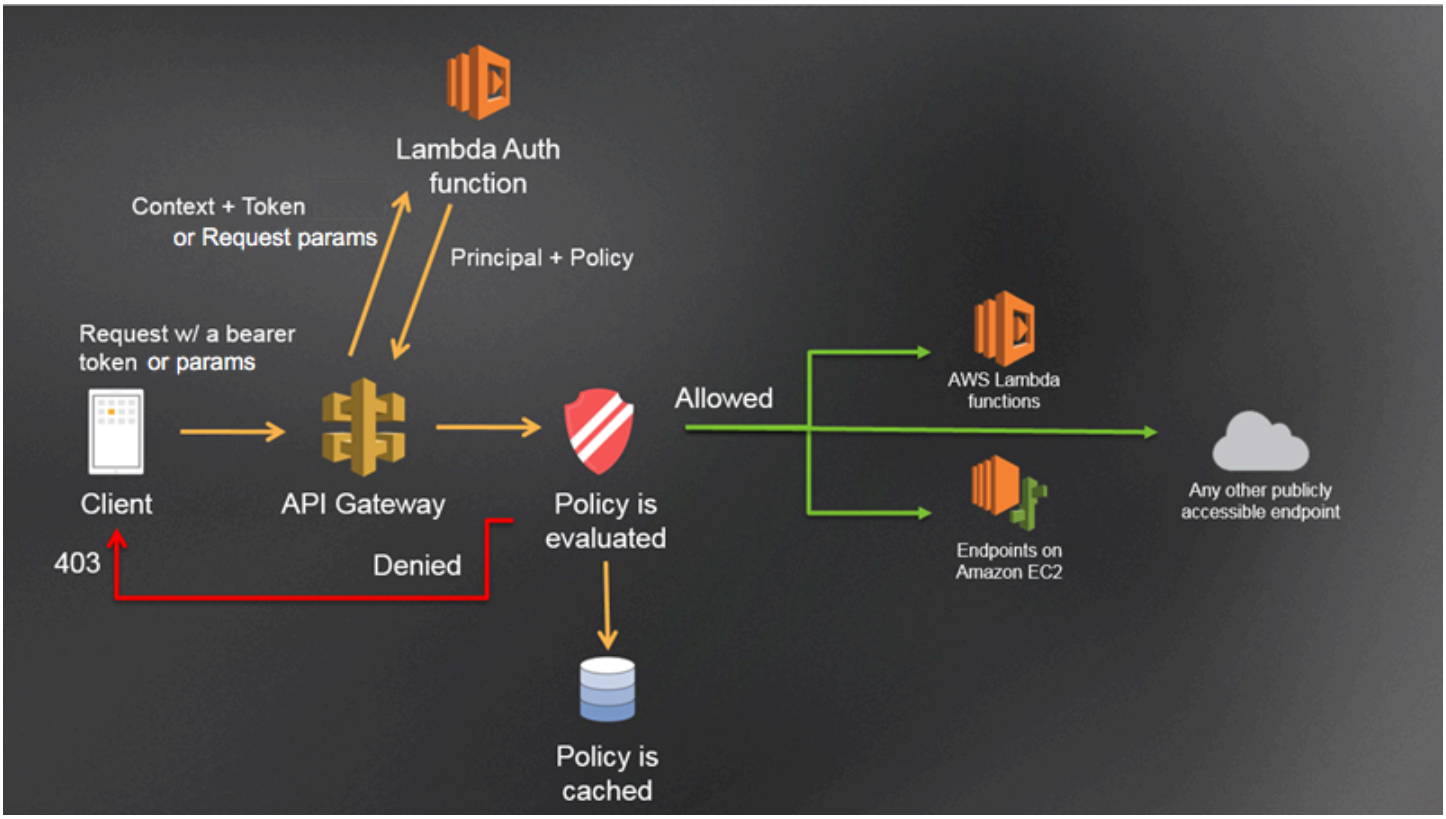
API Gateway-Lambda-Genehmiger verwenden

Verwenden Sie einen Lambda-Autorisierer (früher bekannt als benutzerdefinierter Autorisierer), um den Zugriff auf Ihre API zu kontrollieren. Wenn ein Client die Methode Ihrer API anfordert, ruft API Gateway Ihren Lambda-Authorizer auf. Der Lambda-Autorisierer verwendet die Identität des Aufrufers als Eingabe und gibt eine IAM-Richtlinie als Ausgabe zurück.

Verwenden Sie einen Lambda-Autorisierer, um ein benutzerdefiniertes Autorisierungsschema zu implementieren. Ihr Schema kann Anforderungsparameter verwenden, um die Identität des Aufrufers zu ermitteln, oder eine Bearer-Token-Authentifizierungsstrategie wie OAuth oder SAML verwenden. Erstellen Sie einen Lambda-Authorizer in der API Gateway REST API-Konsole, indem Sie das oder ein AWS CLI AWS SDK verwenden.

Autorisierungsworkflow für Lambda-Autorisierer

Das folgende Diagramm zeigt den Autorisierungsworkflow für einen Lambda-Autorisierer.



API Gateway-Lambda-Autorisierungs-Workflow

1. Der Client ruft eine Methode auf einer API Gateway auf und übergibt dabei ein Bearer-Token oder Anforderungsparameter.
2. API Gateway prüft, ob die Methodenanforderung mit einem Lambda-Authorizer konfiguriert ist. Ist dies der Fall, ruft API Gateway die Lambda-Funktion auf.
3. Die Lambda-Funktion authentifiziert den Anrufer. Die Funktion kann sich auf folgende Weise authentifizieren:
   - Indem Sie einen OAuth-Anbieter anrufen, um ein OAuth-Zugriffstoken zu erhalten.
   - Indem Sie einen SAML-Anbieter anrufen, um eine SAML-Assertion zu erhalten.
   - Durch Generieren einer IAM-Richtlinie auf der Grundlage der Werte der Anforderungsparameter.
   - Durch das Abrufen von Anmeldeinformationen aus einer Datenbank.
4. Die Lambda-Funktion gibt eine IAM-Richtlinie und eine Prinzipal-ID zurück. Wenn die Lambda-Funktion diese Informationen nicht zurückgibt, schlägt der Aufruf fehl.
5. API Gateway bewertet die IAM-Richtlinie.

- Wenn der Zugriff verweigert wird, gibt API Gateway einen geeigneten HTTP-Statuscode zurück, wie z. B. `403 ACCESS_DENIED`.
- Wenn der Zugriff erlaubt ist, ruft API Gateway die Methode auf.

  Wenn Sie das Autorisierungs-Caching aktivieren, speichert API Gateway die Richtlinie im Cache, sodass die Lambda-Autorisierungsfunktion nicht erneut aufgerufen wird.

Sie können die oder die Gateway-Antworten anpassen. `403 ACCESS_DENIED` `401 UNAUTHORIZED` Weitere Informationen hierzu finden Sie unter the section called “Gateway-Antworten”.

Auswahl eines Lambda-Autorisierungstyps

Es gibt zwei Arten von Lambda-Genehmigern:

Parameterbasierten Lambda-Authorizer (Authorizer) anfordern **`REQUEST`**

Ein `REQUEST` Autorisierer empfängt die Identität des Aufrufers in einer Kombination aus Headern, Abfragezeichenfolgenparametern und Variablen. `stageVariables$context` Sie können einen `REQUEST` Autorisierer verwenden, um detaillierte Richtlinien zu erstellen, die auf den Informationen aus mehreren Identitätsquellen basieren, z. B. den Variablen und den Kontextvariablen. `$context.path` `$context.httpMethod`

Wenn Sie das Autorisierungs-Caching für einen `REQUEST` Autorisierer aktivieren, überprüft API Gateway, ob alle angegebenen Identitätsquellen in der Anfrage vorhanden sind. Wenn eine angegebene Identitätsquelle fehlt, null oder leer ist, gibt API Gateway eine `401 Unauthorized` HTTP-Antwort zurück, ohne die Lambda-Autorisierungsfunktion aufzurufen. Wenn mehrere Identitätsquellen definiert sind, werden sie alle verwendet, um den Cache-Schlüssel des Autorisierers abzuleiten, wobei die Reihenfolge beibehalten wird. Sie können einen detaillierten Cache-Schlüssel definieren, indem Sie mehrere Identitätsquellen verwenden.

Wenn Sie einen Teil des Cache-Schlüssels ändern und Ihre API erneut bereitstellen, verwirft der Autorisierer das zwischengespeicherte Richtliniendokument und generiert ein neues.

Wenn Sie das Autorisierungs-Caching für einen `REQUEST` Autorisierer deaktivieren, leitet API Gateway die Anfrage direkt an die Lambda-Funktion weiter.

Tokenbasierter Lambda-Autorisierer (Autorisierer) **`TOKEN`**

Ein `TOKEN` Autorisierer empfängt die Identität des Aufrufers in einem Trägertoken, z. B. einem JSON Web Token (JWT) oder einem OAuth-Token.

Wenn Sie das Autorisierungs-Caching für einen `TOKEN` Authorizer aktivieren, wird der in der Token-Quelle angegebene Header-Name zum Cache-Schlüssel.

Darüber hinaus können Sie die Token-Validierung verwenden, um eine RegEx Anweisung einzugeben. API Gateway führt eine erste Validierung des Eingabe-Tokens anhand dieses Ausdrucks durch und ruft bei erfolgreicher Validierung die Lambda-Autorisierungsfunktion auf. Dies trägt dazu bei, die Anzahl der Aufrufe Ihrer API zu reduzieren.

Die `IdentityValidationExpression` Eigenschaft wird nur für `TOKEN` Autorisierer unterstützt. Weitere Informationen finden Sie unter the section called “`x-amazon-apigateway-authorizer`”.

> **Note**
>
> Wir empfehlen, dass Sie einen `REQUEST` Autorisierer verwenden, um den Zugriff auf Ihre API zu kontrollieren. Sie können den Zugriff auf Ihre API auf der Grundlage mehrerer Identitätsquellen steuern, wenn Sie einen `REQUEST` Autorisierer verwenden, im Vergleich zu einer einzigen Identitätsquelle, wenn Sie einen `TOKEN` Autorisierer verwenden. Darüber hinaus können Sie Cache-Schlüssel mithilfe mehrerer Identitätsquellen für einen `REQUEST` Autorisierer trennen.

Beispiel für eine **`REQUEST`** Autorisierungs-Lambda-Funktion

Der folgende Beispielcode erstellt eine Lambda-Autorisierungsfunktion, die eine Anfrage zulässt, wenn der vom Client bereitgestellte `HeaderAuth1` Header, der `QueryString1` Abfrageparameter und die Stage-Variable `StageVar1` alle den angegebenen Werten von `headerValue1queryValue1`, bzw. entsprechen. `stageValue1`

Node.js

```
   // A simple request-based authorizer example to demonstrate how to use request
    // parameters to allow or deny a request. In this example, a request is
    // authorized if the client-supplied HeaderAuth1 header, QueryString1
    // query parameter, and stage variable of StageVar1 all match
    // specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
    // respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));
```

```
    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
    condition.IpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    }  else {
        callback("Unauthorized");
    }
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
```

```
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    pathParameters = event['pathParameters']
```

```
    stageVariables = event['stageVariables']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    restApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    method = apiGatewayArnTmp[2]
    resource = '/'

    if (apiGatewayArnTmp[3]):
        resource += apiGatewayArnTmp[3]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}

    if (headers['HeaderAuth1'] == "headerValue1" and
 queryStringParameters['QueryString1'] == "queryValue1" and
 stageVariables['StageVar1'] == "stageValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
        print('unauthorized')
        raise Exception('Unauthorized') # Return a 401 Unauthorized response
        return 'unauthorized'

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
```

```
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

In diesem Beispiel prüft die Lambda-Autorisierungsfunktion die Eingabeparameter und verhält sich wie folgt:

- Wenn alle erforderlichen Parameterwerte mit den erwarteten Werten übereinstimmen, gibt die Genehmiger-Funktion eine `200 OK`-HTTP-Antwort und eine IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung wird erfolgreich ausgeführt:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
```

```
}
```

- Andernfalls gibt die Autorisierungsfunktion eine `401 Unauthorized` HTTP-Antwort zurück und die Methodenanforderung schlägt fehl.

Zusätzlich zur Rückgabe einer IAM-Richtlinie muss die Lambda-Funktion des Genehmigers auch die Prinzipal-ID des Aufrufers zurückgeben. Optional kann sie ein `context` Objekt zurückgeben, das zusätzliche Informationen enthält, die an das Integrations-Backend übergeben werden können. Weitere Informationen finden Sie unter Ausgabe von einem API Gateway Lambda Authorizer.

Im Produktionscode müssen Sie möglicherweise den Benutzer authentifizieren, bevor Sie die Autorisierung erteilen. Sie können der Lambda-Funktion Authentifizierungslogik hinzufügen, indem Sie einen Authentifizierungsanbieter aufrufen, wie in der Dokumentation für diesen Anbieter beschrieben.

Beispiel für eine **`TOKEN`** Autorisierungs-Lambda-Funktion

Der folgende Beispielcode erstellt eine `TOKEN` Lambda-Autorisierungsfunktion, die es einem Aufrufer ermöglicht, eine Methode aufzurufen, wenn der vom Client bereitgestellte Tokenwert lautet. `allow` Der Aufrufer darf die Anfrage nicht aufrufen, wenn der Tokenwert `deny` Wenn der Tokenwert `unauthorized` oder eine leere Zeichenfolge ist, gibt die Autorisierungsfunktion eine Antwort zurück. `401 UNAUTHORIZED`

Node.js

```
// A simple token-based authorizer example to demonstrate how to use an
 authorization token
// to allow or deny a request. In this example, the caller named 'user' is allowed
 to invoke
// a request if the client-supplied token value is 'allow'. The caller is not
 allowed to invoke
// the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
// string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
// the authorizer returns an HTTP 500 status code.
// Note that token values are case-sensitive.

export const handler =  function(event, context, callback) {
    var token = event.authorizationToken;
    switch (token) {
```

```
        case 'allow':
            callback(null, generatePolicy('user', 'Allow', event.methodArn));
            break;
        case 'deny':
            callback(null, generatePolicy('user', 'Deny', event.methodArn));
            break;
        case 'unauthorized':
            callback("Unauthorized");   // Return a 401 Unauthorized response
            break;
        default:
            callback("Error: Invalid token"); // Return a 500 Invalid token response
    }
};

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    var authResponse = {};

    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17';
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke';
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }

    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}
```

Python

```
# A simple token-based authorizer example to demonstrate how to use an authorization
 token
# to allow or deny a request. In this example, the caller named 'user' is allowed to
 invoke
# a request if the client-supplied token value is 'allow'. The caller is not allowed
 to invoke
# the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
# string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
# the authorizer returns an HTTP 500 status code.
# Note that token values are case-sensitive.

import json


def lambda_handler(event, context):
    token = event['authorizationToken']
    if token == 'allow':
        print('authorized')
        response = generatePolicy('user', 'Allow', event['methodArn'])
    elif token == 'deny':
        print('unauthorized')
        response = generatePolicy('user', 'Deny', event['methodArn'])
    elif token == 'unauthorized':
        print('unauthorized')
        raise Exception('Unauthorized')  # Return a 401 Unauthorized response
        return 'unauthorized'
    try:
        return json.loads(response)
    except BaseException:
        print('unauthorized')
        return 'unauthorized'  # Return a 500 error


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
```

```
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument
    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }
    authResponse_JSON = json.dumps(authResponse)
    return authResponse_JSON
```

Wenn in diesem Beispiel die API eine Methodenanfrage empfängt, übergibt API Gateway das Quell-Token an diese Lambda-Genehmigerfunktion im Attribut `event.authorizationToken`. Die Lambda-Genehmigerfunktion liest das Token und verhält sich wie folgt:

- Wenn der Token-Wert `allow` lautet, gibt die Genehmigerfunktion eine `200 OK`-HTTP-Antwort und eine IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung wird erfolgreich ausgeführt:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Wenn der Token-Wert `deny` lautet, gibt die Genehmigerfunktion eine `200 OK`-HTTP-Antwort und eine `Deny`-IAM-Richtlinie zurück (wie nachfolgend gezeigt), und die Methodenanforderung schlägt fehl:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Action": "execute-api:Invoke",
      "Effect": "Deny",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

> **Note**
>
> Außerhalb der Testumgebung gibt API Gateway eine `403 Forbidden` HTTP-Antwort zurück und die Methodenanforderung schlägt fehl.

- Wenn der Token-Wert `unauthorized` lautet oder eine leere Zeichenfolge ist, gibt die Genehmigerfunktion eine `401 Unauthorized`-HTTP-Antwort zurück, und der Methodenaufruf schlägt fehl.
- Wenn das Token anders lautet, erhält der Client eine `500 Invalid token`-Antwort, und der Methodenaufruf schlägt fehl.

Zusätzlich zur Rückgabe einer IAM-Richtlinie muss die Lambda-Funktion des Genehmigers auch die Prinzipal-ID des Aufrufers zurückgeben. Optional kann es ein `context` Objekt zurückgeben, das zusätzliche Informationen enthält, die an das Integrations-Backend übergeben werden können. Weitere Informationen finden Sie unter Ausgabe von einem API Gateway Lambda Authorizer.

Im Produktionscode müssen Sie möglicherweise den Benutzer authentifizieren, bevor Sie die Autorisierung erteilen. Sie können der Lambda-Funktion Authentifizierungslogik hinzufügen, indem Sie einen Authentifizierungsanbieter aufrufen, wie in der Dokumentation für diesen Anbieter beschrieben.

Zusätzliche Beispiele für Lambda-Autorisierungsfunktionen

Die folgende Liste enthält weitere Beispiele für Lambda-Autorisierungsfunktionen. Sie können eine Lambda-Funktion in demselben Konto oder in einem anderen Konto erstellen, von dem aus Sie Ihre API erstellt haben.

Für die vorherigen Lambda-Beispielfunktionen können Sie die integrierten Funktionen verwenden AWSLambdaBasicExecutionRole, da diese Funktionen keine anderen AWS Dienste aufrufen. Wenn Ihre Lambda-Funktion andere AWS Dienste aufruft, müssen Sie der Lambda-Funktion eine IAM-

Ausführungsrolle zuweisen. Folgen Sie beim Erstellen einer Rolle den Anweisungen unter AWS Lambda -Ausführungsrolle.

Zusätzliche Lambda-Autorisierungsfunktionen als Beispiel

- Eine Beispielanwendung finden Sie unter Open Banking Brazil — Authorization Samples on. GitHub
- Weitere Lambda-Beispielfunktionen finden Sie unter aws-apigateway-lambda-authorizer-blueprints on. GitHub
- Sie können einen Lambda-Autorisierer erstellen, der Benutzer mithilfe von Amazon Cognito Cognito-Benutzerpools authentifiziert und Anrufer anhand eines Richtlinienspeichers mithilfe verifizierter Berechtigungen autorisiert. Weitere Informationen finden Sie unter Erstellen eines Richtlinienspeichers mit einer verbundenen API und einem verbundenen Identitätsanbieter im Amazon Verified Permissions User Guide.
- Die Lambda-Konsole stellt einen Python-Blueprint bereit, den Sie verwenden können, indem Sie Blueprint verwenden und dann Blueprint auswählen. api-gateway-authorizer-python

Einen Lambda-Autorisierer konfigurieren

Nachdem Sie eine Lambda-Funktion erstellt haben, konfigurieren Sie die Lambda-Funktion als Autorisierer für Ihre API. Anschließend konfigurieren Sie Ihre Methode so, dass sie Ihren Lambda-Authorizer aufruft, um festzustellen, ob ein Aufrufer Ihre Methode aufrufen kann. Sie können eine Lambda-Funktion in demselben Konto oder in einem anderen Konto erstellen, von dem aus Sie Ihre API erstellt haben.

Sie können Ihren Lambda-Authorizer mit integrierten Tools in der API Gateway Gateway-Konsole oder mithilfe von Postman testen. Anweisungen zur Verwendung von Postman zum Testen Ihrer Lambda-Autorisierungsfunktion finden Sie unter. the section called “Aufruf einer API mit Lambda-Genehmigern”

Konfigurieren Sie einen Lambda-Autorisierer (Konsole)

Das folgende Verfahren zeigt, wie Sie einen Lambda-Authorizer in der API Gateway REST API-Konsole erstellen. Weitere Informationen zu den verschiedenen Typen von Lambda-Autorisierern finden Sie unter. the section called “Auswahl eines Lambda-Autorisierungstyps”

REQUEST authorizer

So konfigurieren Sie einen **`REQUEST`** Lambda-Autorisierer

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine API und dann Authorizers aus.
3. Wählen Sie Genehmiger erstellen aus.
4. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
5. Wählen Sie als Genehmiger-Typ Lambda aus.
6. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Autorisierungsfunktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
7. Lassen Sie das Feld Lambda invoke role leer, damit die API Gateway REST API-Konsole eine ressourcenbasierte Richtlinie festlegen kann. Die Richtlinie gewährt API Gateway Gateway-Berechtigungen zum Aufrufen der Lambda-Autorisierungsfunktion. Sie können auch den Namen einer IAM-Rolle eingeben, damit API Gateway die Lambda-Autorisierungsfunktion aufrufen kann. Ein Beispiel für eine Rolle finden Sie unter. Übernehmbare IAM-Rolle erstellen
8. Wählen Sie für Lambda-Ereignisnutzlast die Option Anfrage.
9. Wählen Sie für Identitätsquellen-Typ einen Parametertyp aus. Die unterstützten Parametertypen sind `Header`, `Query string`, `Stage variable` und `Context`. Um weitere Identitätsquellen hinzuzufügen, wählen Sie Parameter hinzufügen.
10. Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn das Policy-Caching aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert.

    Wenn Sie das Caching aktivieren, muss Ihr Autorisierer eine Richtlinie zurückgeben, die für alle Methoden in einer API gilt. Um eine methodenspezifische Richtlinie durchzusetzen, verwenden Sie die Kontextvariablen und. `$context.path` `$context.httpMethod`
11. Wählen Sie Autorisierer erstellen.

TOKEN authorizer

So konfigurieren Sie einen **`TOKEN`** Lambda-Autorisierer

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine API und dann Authorizers aus.
3. Wählen Sie Genehmiger erstellen aus.
4. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
5. Wählen Sie als Genehmiger-Typ Lambda aus.
6. Wählen Sie für Lambda-Funktion den Ort aus, AWS-Region an dem Sie Ihre Lambda-Autorisierungsfunktion erstellt haben, und geben Sie dann den Funktionsnamen ein.
7. Lassen Sie das Feld Lambda invoke role leer, damit die API Gateway REST API-Konsole eine ressourcenbasierte Richtlinie festlegen kann. Die Richtlinie gewährt API Gateway Gateway-Berechtigungen zum Aufrufen der Lambda-Autorisierungsfunktion. Sie können auch den Namen einer IAM-Rolle eingeben, damit API Gateway die Lambda-Autorisierungsfunktion aufrufen kann. Ein Beispiel für eine Rolle finden Sie unter. Übernehmbare IAM-Rolle erstellen
8. Wählen Sie für Lambda-Ereignisnutzlast die Option Token aus.
9. Geben Sie unter Token-Quelle den Header-Namen ein, der das Genehmigungstoken enthält. Der Aufrufer muss einen Header mit diesem Namen angeben, um das Autorisierungstoken an den Lambda-Autorisierer zu senden.
10. (Optional) Geben Sie für die Token-Validierung eine Anweisung ein. RegEx API Gateway führt eine erste Überprüfung des eingegebenen Token für diesen Ausdruck an und ruft nach der erfolgreichen Validierung den Genehmiger auf.
11. Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn das Richtlinien-Caching aktiviert ist, wird der unter Token-Quelle angegebene Header-Name zum Cache-Schlüssel. Wenn das Policy-Caching aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert.

    Wenn Sie das Caching aktivieren, muss Ihr Autorisierer eine Richtlinie zurückgeben, die für alle Methoden in einer API gilt. Um eine methodenspezifische Richtlinie durchzusetzen, können Sie das Autorisierungs-Caching deaktivieren.
12. Wählen Sie Autorisierer erstellen.

Nachdem Sie Ihren Lambda-Authorizer erstellt haben, können Sie ihn testen. Das folgende Verfahren zeigt, wie Sie Ihren Lambda-Authorizer testen.

REQUEST authorizer

Um einen **`REQUEST`** Lambda-Autorisierer zu testen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie den Namen Ihres Autorisierers aus.
3. Geben Sie unter Autorisierer testen einen Wert für Ihre Identitätsquelle ein.

   Wenn Sie den verwendenthe section called “Beispiel für eine REQUEST Autorisierungs-Lambda-Funktion”, gehen Sie wie folgt vor:

   a. Wählen Sie Header, geben Sie **`headerValue1`** ein und wählen Sie dann Parameter hinzufügen aus.
   b. Wählen Sie unter Identitätsquellen-Typ die Option Abfragezeichenfolge, geben Sie **`queryValue1`** ein und wählen Sie dann Parameter hinzufügen aus.
   c. Wählen Sie unter Identitätsquellen-Typ die Option Stufenvariable aus und geben Sie **`stageValue1`** ein.

   Sie können die Kontextvariablen für den Testaufruf nicht ändern, aber Sie können die API Gateway Authorizer-Testereignisvorlage für Ihre Lambda-Funktion ändern. Anschließend können Sie Ihre Lambda-Autorisierungsfunktion mit geänderten Kontextvariablen testen. Weitere Informationen finden Sie im AWS Lambda Entwicklerhandbuch unter Testen von Lambda-Funktionen in der Konsole.
4. Wählen Sie Genehmiger testen aus.

TOKEN authorizer

Um einen **`TOKEN`** Lambda-Autorisierer zu testen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie den Namen Ihres Autorisierers aus.
3. Geben Sie unter Autorisierer testen einen Wert für Ihr Token ein.

Wenn Sie den verwendenthe section called “Beispiel für eine TOKEN Autorisierungs-Lambda-Funktion”, gehen Sie wie folgt vor:

- Geben Sie für das authorizationToken ein. **`allow`**

4. Wählen Sie Genehmiger testen aus.

   Wenn Ihr Lambda-Autorisierer eine Anfrage in der Testumgebung erfolgreich ablehnt, antwortet der Test mit einer `200 OK` HTTP-Antwort. Außerhalb der Testumgebung gibt API Gateway jedoch eine `403 Forbidden` HTTP-Antwort zurück und die Methodenanforderung schlägt fehl.

Einen Lambda-Autorisierer konfigurieren ()AWS CLI

Der folgende create-authorizer-Befehl zeigt, wie Sie einen Lambda-Authorizer mit dem erstellen. AWS CLI

REQUEST authorizer

Das folgende Beispiel erstellt einen `REQUEST` Authorizer und verwendet den `Authorizer` Header und `accountId` die Kontextvariable als Identitätsquellen:

```
aws apigateway create-authorizer \
    --rest-api-id 1234123412 \
    --name 'First_Request_Custom_Authorizer' \
    --type REQUEST \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization,context.accountId' \
    --authorizer-result-ttl-in-seconds 300
```

TOKEN authorizer

Das folgende Beispiel erstellt einen `TOKEN` Authorizer und verwendet den `Authorization` Header als Identitätsquelle:

```
aws apigateway create-authorizer \
    --rest-api-id 1234123412 \
    --name 'First_Token_Custom_Authorizer' \
    --type TOKEN \
```

```
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization' \
    --authorizer-result-ttl-in-seconds 300
```

Nachdem Sie Ihren Lambda-Authorizer erstellt haben, können Sie ihn testen. Der folgende test-invoke-authorizerBefehl zeigt, wie Sie Ihren Lambda-Authorizer testen:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \
   --authorizer-id efg1234 \
   --headers Authorization='Value'
```

Konfigurieren Sie eine Methode zur Verwendung eines Lambda-Autorisierers (Konsole)

Nachdem Sie Ihren Lambda-Authorizer konfiguriert haben, müssen Sie ihn an eine Methode für Ihre API anhängen.

So konfigurieren Sie eine API-Methode für die Verwendung eines Lambda-Genehmigers:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine API aus.
3. Wählen Sie Ressourcen und wählen Sie dann eine neue Methode oder wählen Sie eine vorhandene Methode.
4. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
5. Wählen Sie für Genehmiger aus dem Drop-down-Menü den Lambda-Genehmiger aus, den Sie gerade erstellt haben.
6. (Optional) Wenn Sie das Genehmigungstoken an das Backend übergeben möchten, wählen Sie HTTP-Anforderungsheader aus. Wählen Sie Header hinzufügen und fügen Sie dann den Namen des Genehmigungsheaders hinzu. Geben Sie unter Name den Header-Namen ein, der dem Token-Quellnamen entspricht, den Sie bei der Erstellung des Lambda-Autorisierers für die API angegeben haben. Dieser Schritt gilt nicht für `REQUEST`-Genehmiger.
7. Wählen Sie Speichern.
8. Wählen Sie Deploy API aus, um die API für eine Stufe bereitzustellen. Für einen `REQUEST`-Genehmiger mit Stufenvariablen müssen Sie auch die erforderlichen Stufenvariablen definieren und ihre Werte auf der Seite Stufen angeben.

Konfigurieren Sie die Methode einer API für die Verwendung eines Lambda-Autorisierers ()AWS CLI

Nachdem Sie Ihren Lambda-Authorizer konfiguriert haben, müssen Sie ihn an eine Methode für Ihre API anhängen. Sie können eine neue Methode erstellen oder einen Patch-Vorgang verwenden, um einen Autorisierer an eine bestehende Methode anzuhängen.

Der folgende Befehl put-method zeigt, wie eine neue Methode erstellt wird, die einen Lambda-Authorizer verwendet:

```
aws apigateway put-method --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method PUT \
  --authorization-type CUSTOM \
  --authorizer-id efg1234
```

Der folgende Befehl update-method zeigt, wie eine vorhandene Methode aktualisiert wird, um einen Lambda-Authorizer zu verwenden:

```
aws apigateway update-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --patch-operations op="replace",path="/authorizationType",value="CUSTOM"
 op="replace",path="/authorizerId",value="efg1234"
```

Eingabe an einen Amazon API Gateway-Lambda-Genehmiger

`TOKEN`-Eingabeformat

Für einen Lambda-Genehmiger (ehemals als benutzerdefinierter Genehmiger bezeichnet) des Typs `TOKEN` müssen Sie einen benutzerdefinierten Header als Token Source (Token-Quelle) angeben, wenn Sie den Genehmiger für Ihre API konfigurieren. Der API-Client muss das erforderliche Autorisierungs-Token in diesem Header in der eingehenden Anforderung übergeben. Beim Empfang der eingehenden Methodenanfrage extrahiert API Gateway das Token aus dem benutzerdefinierten Header. Anschließend gibt es das Token als `authorizationToken`-Eigenschaft des `event`-Objekts der Lambda-Funktion zurück, zusätzlich zum Methode-ARN als `methodArn`-Eigenschaft:

```
{
    "type":"TOKEN",
    "authorizationToken":"{caller-supplied-token}",
```

```
    "methodArn":"arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}/[{child-resources}]]"
}
```

In diesem Beispiel gibt die `type`-Eigenschaft den Genehmigertyp an, wobei es sich um einen `TOKEN` Genehmiger handelt. Das *`{caller-supplied-token}`* stammt aus dem Autorisierungs-Header einer Client-Anforderung und kann eine beliebige Zeichenfolge sein. `methodArn` ist der ARN der eingehenden Methodenanfrage und wird von API Gateway entsprechend der Konfiguration des Lambda-Genehmigers aufgefüllt.

`REQUEST`-Eingabeformat

Für einen Lambda-Genehmiger vom Typ `REQUEST` übergibt API Gateway die Anfrageparameter als Teil des `event`-Objekts an die Lambda-Genehmigerfunktion. Die Anforderungsparameter umfassen Header, Pfad, Parameter für Abfragezeichenfolgen, Stufenvariablen und einige der Anforderungskontextvariablen. Der API-Aufrufer können die Pfadparameter, Header und Abfragezeichenfolgenparameter festlegen. Der API-Entwickler muss die Stage-Variablen während der API-Bereitstellung festlegen, und API Gateway stellt den Anfragekontext zur Laufzeit zur Verfügung.

> **Note**
>
> Pfadparameter können als Anfrageparameter an die Lambda-Genehmigerfunktion übergeben werden, aber sie können nicht als Identitätsquellen verwendet werden.

Das folgende Beispiel zeigt eine Eingabe für einen `REQUEST` Genehmiger für eine API-Methode (`GET /request`) mit einer Proxy-Integration:

```
{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
  "resource": "/request",
  "path": "/request",
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
```

```
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

`requestContext` ist eine Zuordnung von Schlüssel-Wert-Paaren und entspricht der Variablen $ Kontext. Das Ergebnis hängt von der API ab.

API Gateway fügt der Map möglicherweise neue Schlüssel hinzu. Weitere Informationen über die Eingabe von Lambda-Funktionen bei der Lambda-Proxy-Integration finden Sie unter Eingabeformat einer Lambda-Funktion für die Proxy-Integration.

Ausgabe von einem API Gateway Lambda Authorizer

Die Ausgabe einer Lambda-Genehmigerfunktion ist ein wörterbuchähnliches Objekt, das die Prinzipal-ID (`principalId`) und ein Richtliniendokument (`policyDocument`) mit einer Liste Richtlinienanweisungen umfassen muss. Die Ausgabe kann auch eine `context`-Zuweisung mit Schlüssel-Wert-Paaren umfassen. Wenn die API einen Nutzungsplan verwendet (`apiKeySource` ist auf `AUTHORIZER` gesetzt), muss die Lambda-Genehmigerfunktion einen der API-Schlüssel des Nutzungsplans als `usageIdentifierKey`-Eigenschaftswert ausgeben.

Hier ein Beispiel für diese Ausgabe.

```
{
  "principalId": "yyyyyyyy", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-
api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "stringKey": "value",
    "numberKey": "1",
    "booleanKey": "true"
  },
  "usageIdentifierKey": "{api-key}"
}
```

Hier legt eine Richtlinienanweisung fest, ob dem API Gateway-Ausführungs-Service der Aufruf (`Action`) der angegebenen API-Methode (`Resource`) erlaubt oder verweigert wird (`Effect`). Sie können einen Platzhalter (*) verwenden, um einen Ressourcentyp (Methode) anzugeben. Weitere Informationen zum Festlegen von gültigen Richtlinien für das Aufrufen einer API finden Sie unter Anweisungsreferenz für IAM-Richtlinien zur Ausführung der API in API Gateway.

Für Methoden-ARNs mit Autorisierung, z. B. `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-`

resources}]] beträgt die maximale Länge 1 600 Bytes. Die Pfadparameterwerte, deren Größe zur Laufzeit ermittelt wird, können dazu führen, dass die Länge des ARN dieses Limit übersteigt. Wenn dies geschieht, empfängt der API-Client die Antwort `414 Request URI too long`.

Darüber hinaus ist der Ressourcen-ARN, wie in der Richtlinienanweisungsausgabe des Genehmigers gezeigt, derzeit auf 512 Zeichen begrenzt. Aus diesem Grund können Sie die URI nicht mit einem längeren JWT-Token in einer Anforderungs-URI verwenden. Übergeben Sie das JWT-Token stattdessen sicher im Anforderungs-Header.

Sie können auf den `principalId`-Wert in einer Mapping-Vorlage unter Verwendung der `$context.authorizer.principalId`-Variablen zugreifen. Dies ist nützlich, wenn Sie den Wert an das Backend übergeben möchten. Weitere Informationen finden Sie unter $contextVariablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch .

Sie können auf den `stringKey`-, `numberKey`- oder `booleanKey`-Wert (z. B. `"value"`, `"1"` oder `"true"`) der `context`-Zuweisung in einer Mapping-Vorlage zugreifen, indem Sie `$context.authorizer.stringKey`, `$context.authorizer.numberKey` bzw. `$context.authorizer.booleanKey` aufrufen. Die zurückgegebenen Werte werden alle in Text umgewandelt. Beachten Sie, dass Sie kein JSON-Objekt oder -Array als gültigen Wert eines Schlüssels in der `context`-Zuweisung festlegen können.

Sie können das `context`-Mapping verwenden, um im Cache abgelegte Anmeldeinformationen vom Genehmiger unter Verwendung einer Mapping-Vorlage für die Integrationsanfrage an das Backend zurückzugeben. Dies ermöglicht dem Backend, durch Nutzung der im Cache gespeicherten Anmeldeinformationen, um die Notwendigkeit für den Zugriff auf die geheimen Schlüssel zu reduzieren und die Autorisierungs-Token für jede Anforderung zu öffnen, eine verbesserte Benutzerumgebung bereitzustellen.

Für die Lambda-Proxy-Integration übergibt API Gateway das `context`-Objekt von einem Lambda-Genehmiger direkt an die Backend-Lambda-Funktion als Teil der Eingabe `event`. Sie können die `context`-Schlüssel-/Wert-Paare in der Lambda-Funktion durch den Aufruf von `$event.requestContext.authorizer.`*`key`* abrufen.

`{api-key}` steht für einen API-Schlüssel im Nutzungsplan der API-Stufe. Weitere Informationen finden Sie unter the section called “Nutzungspläne”.

Im Folgenden wird die Beispielausgabe aus dem Beispiel-Lambda-Genehmiger gezeigt. Die Beispielausgabe enthält eine Richtlinienanweisung zum Blockieren (`Deny`) von `GET` Methodenaufrufen für die `dev` Phase einer API (`ymy8tbxw7b`) eines AWS Accounts (`123456789012`).

```
{
  "principalId": "user",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/GET/"
      }
    ]
  }
}
```

Aufruf einer API mit API Gateway-Lambda-Genehmigern

Nachdem Sie den Lambda-Genehmiger (früher als benutzerdefinierter Genehmiger bezeichnet) konfiguriert und die API bereitgestellt haben, sollten Sie die API mit aktiviertem Lambda-Genehmiger testen. Dazu benötigen Sie einen REST-Client, z. B. cURL oder Postman. In den folgenden Beispielen verwenden wir Postman.

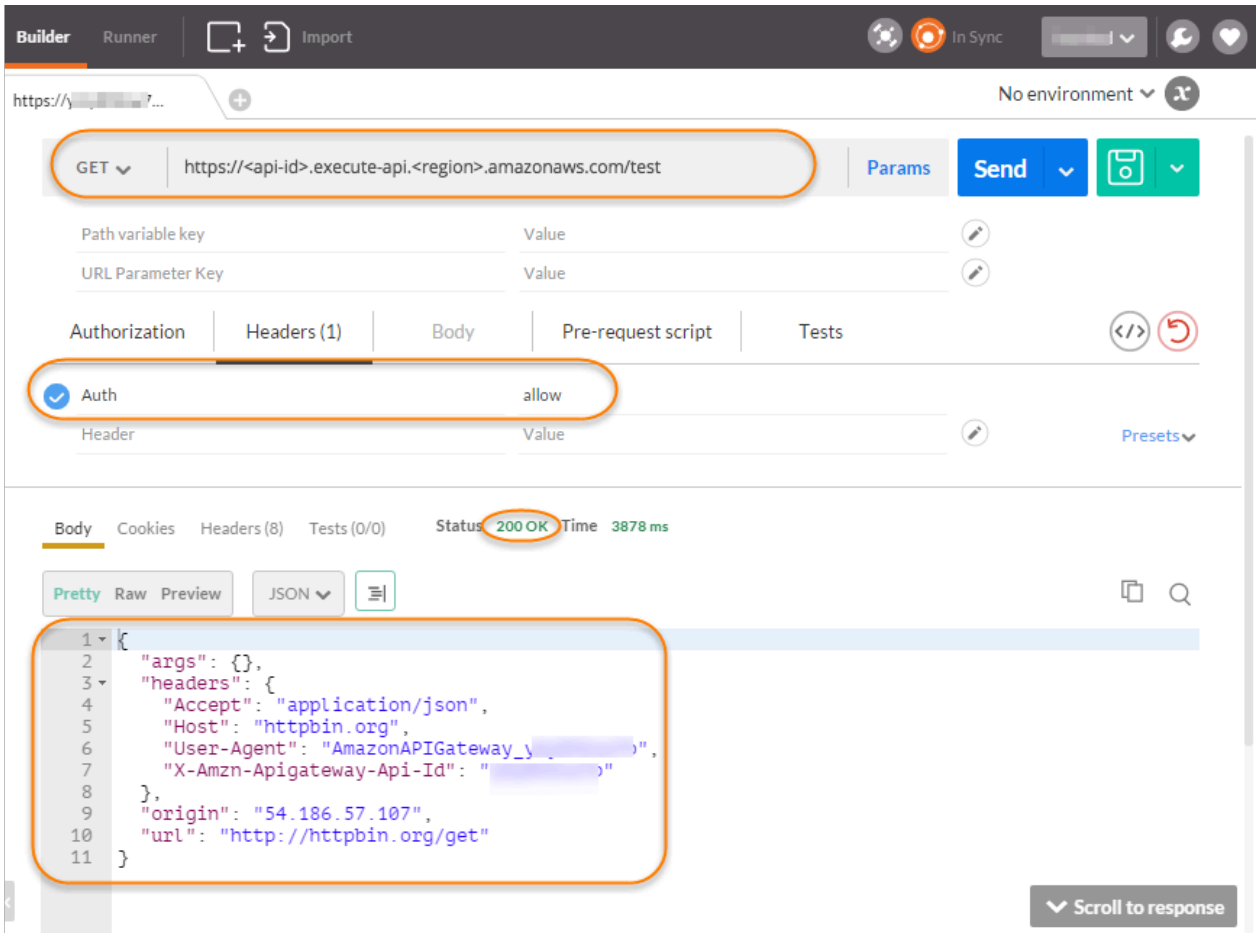
> **Note**
>
> Beim Aufrufen einer autorisiererfähigen Methode protokolliert API Gateway den Aufruf nicht, CloudWatch wenn das erforderliche Token für den **`TOKEN`** Authorizer nicht gesetzt ist, Null ist oder durch den angegebenen Token-Validierungsausdruck ungültig gemacht wird. Ebenso protokolliert API Gateway den Aufruf nicht, CloudWatch wenn eine der erforderlichen Identitätsquellen für den `REQUEST` Autorisierer nicht festgelegt, Null oder leer ist.

Nachfolgend zeigen wir, wie Sie Postman verwenden, um mit einem Lambda-`TOKEN`-Genehmiger die API aufzurufen oder zu testen. Die Methode kann für das Aufrufen einer API mit einem Lambda-`REQUEST`-Genehmiger angewendet werden, wenn Sie den erforderlichen Parameter für Pfad, Header oder Abfragezeichenfolge explizit angeben.

Aufrufen einer API mit dem benutzerdefinierten **`TOKEN`** Genehmiger

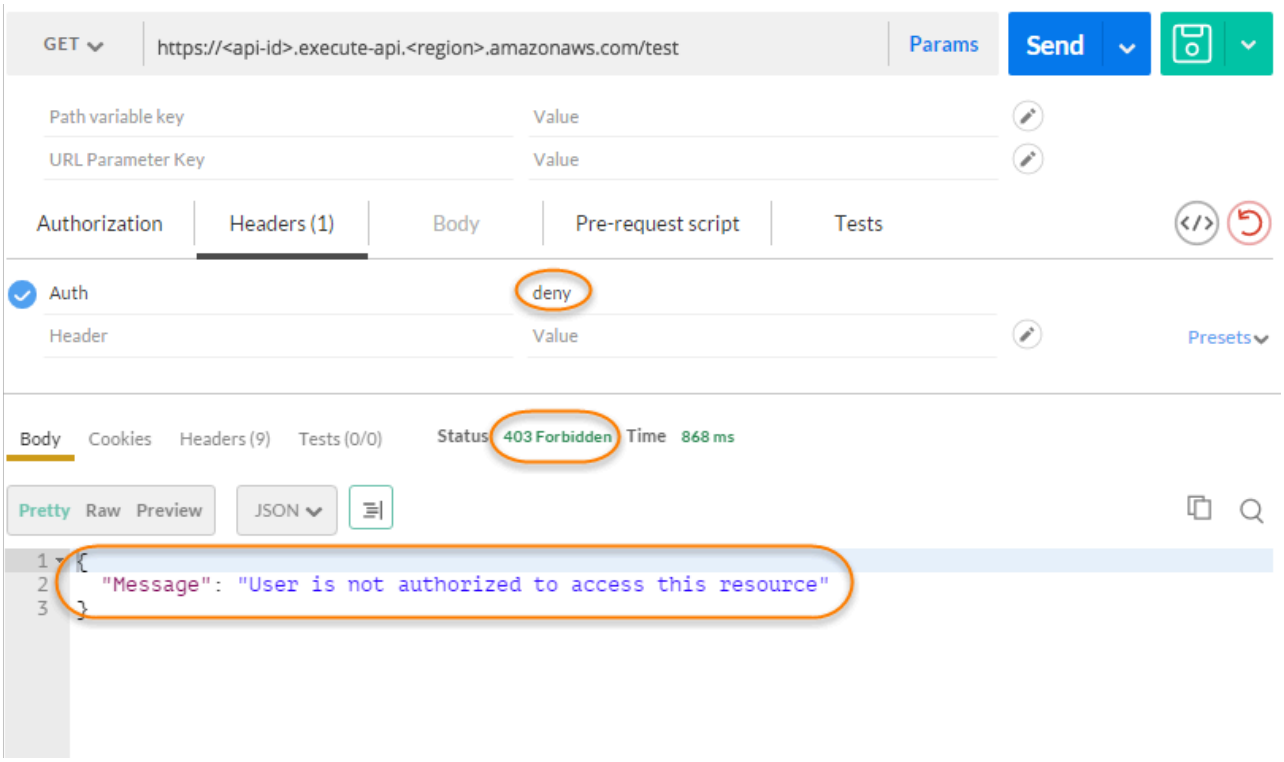
1. Öffnen Sie Postman, wählen Sie die GET-Methode aus und fügen Sie die Invoke URL der API im benachbarten URL-Feld ein.

Fügen Sie den Lambda-Genehmiger-Token-Header hinzu und setzen Sie den Wert auf `allow`. Wählen Sie Send (Senden) aus.



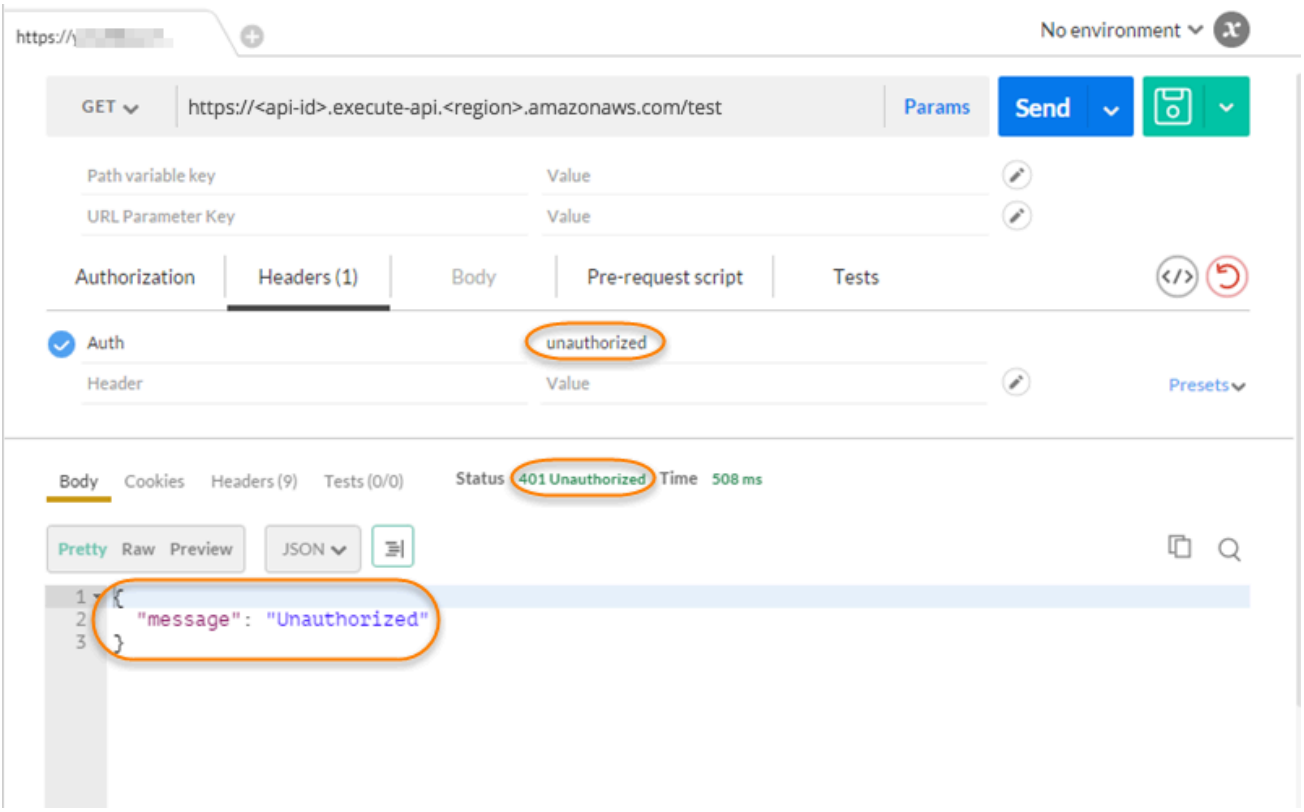
Die Antwort zeigt, dass der API Gateway-Lambda-Genehmiger die Antwort 200 OK zurückgibt und den Aufruf erfolgreich für den Zugriff auf den mit der Methode integrierten HTTP-Endpunkt (http://httpbin.org/get) autorisiert.

2. Ändern Sie in Postman den Lambda-Autorisierungstoken-Header-Wert in `deny`. Wählen Sie Send (Senden) aus.



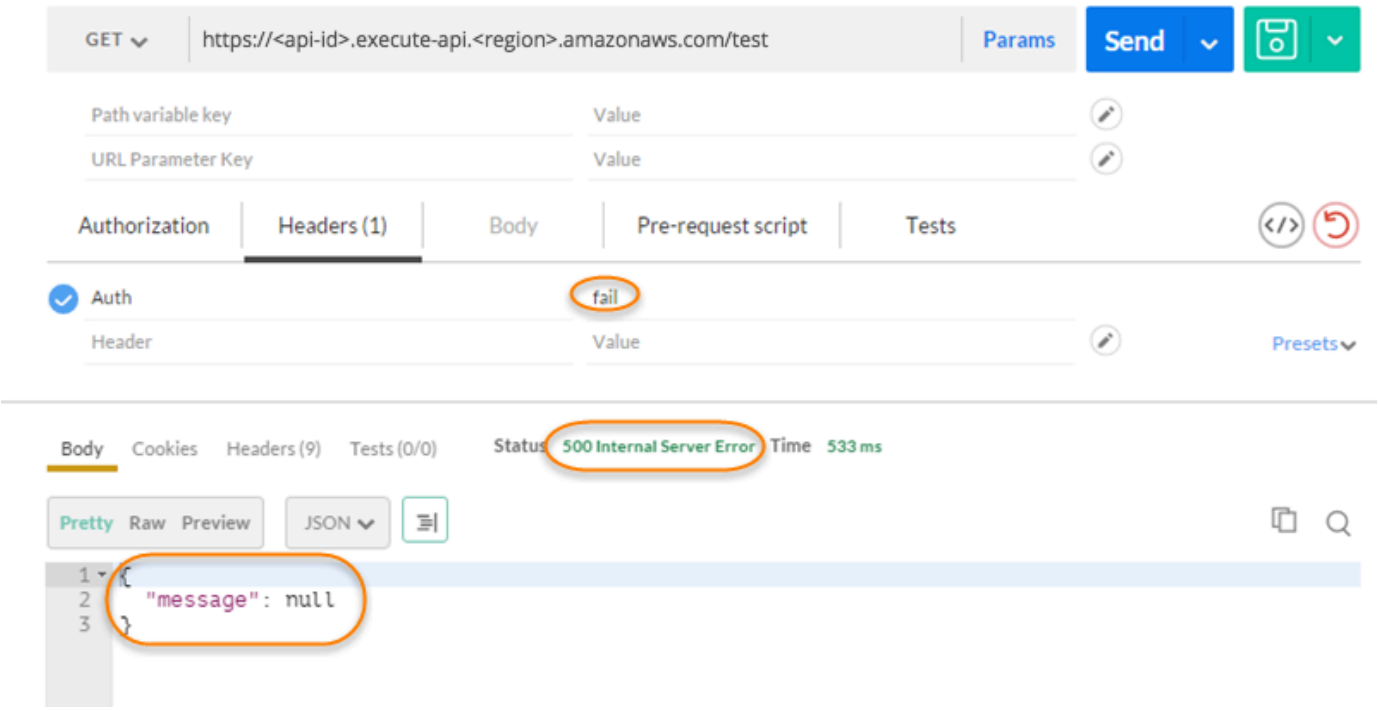
Die Antwort zeigt, dass der API Gateway-Lambda-Genehmiger die Antwort 403 Forbidden zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

3. Ändern Sie in Postman den Wert des Lambda-Autorisierungstoken-Headers in `unauthorized` und wählen Sie Send (Senden).



Die Antwort zeigt, dass API Gateway die Antwort 401 Unauthorized zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

4. Ändern Sie nun den Lambda-Genehmiger-Token-Header-Wert in `fail`. Wählen Sie Send (Senden) aus.



Die Antwort zeigt, dass API Gateway die Antwort 500 Internal Server Error zurückgibt, ohne den Aufruf zum Zugriff auf den HTTP-Endpunkt zu autorisieren.

Kontenübergreifenden Lambda-Genehmiger konfigurieren

Sie können jetzt auch eine AWS Lambda Funktion aus einem anderen AWS Konto als API-Autorisierungsfunktion verwenden. Jedes Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist. Die Lambda-Genehmiger-Funktion unterstützt Bearer-Token-Authentifizierungsstrategien wie OAuth oder SAML. Dies erleichtert die zentrale Verwaltung und gemeinsame Nutzung einer zentralen Lambda-Genehmigerfunktion über mehrere API Gateway-APIs hinweg.

In diesem Abschnitt zeigen wir, wie eine kontoübergreifende Lambda-Genehmigerfunktion mit der Amazon API Gateway-Konsole konfiguriert wird.

Bei diesen Anweisungen wird davon ausgegangen, dass Sie bereits eine API-Gateway-API in einem AWS Konto und eine Lambda-Autorisierungsfunktion in einem anderen Konto haben.

Kontoübergreifenden Lambda-Genehmiger mit der API Gateway-Konsole konfigurieren

Melden Sie sich in der Amazon-API-Gateway-Konsole bei dem Konto an, in dem Ihre API enthalten ist, und gehen Sie wie folgt vor:

1. Wählen Sie Ihre API und dann im Hauptnavigationsbereich Genehmiger.
2. Wählen Sie Genehmiger erstellen aus.
3. Geben Sie unter Name des Genehmigers einen Namen für den Genehmiger ein.
4. Wählen Sie als Genehmiger-Typ Lambda aus.
5. Geben Sie für Lambda-Funktion den vollständigen ARN der Lambda-Genehmigerfunktion aus Ihrem zweiten Konto ein.

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

6. Eine Warnung mit einer `aws lambda add-permission`-Befehlszeichenfolge wird angezeigt. Diese Richtlinie gewährt API Gateway die Berechtigung, die Genehmiger-Lambda-Funktion aufzurufen. Kopieren Sie den Befehl und speichern Sie ihn für später. Führen Sie den Befehl aus, nachdem Sie den Genehmiger erstellt haben.
7. Lassen Sie Lambda-Aufrufrolle leer, damit die API-Gateway-Konsole eine ressourcenbasierte Richtlinie festlegen kann. Die Richtlinie gewährt API Gateway die Berechtigung, die Lambda-Genehmigerfunktion aufzurufen. Sie können auch eine IAM-Rolle eingeben, damit API Gateway die Genehmiger-Lambda-Funktion aufrufen kann. Ein Beispiel für eine solche Rolle finden Sie unter Übernehmbare IAM-Rolle erstellen.
8. Wählen Sie für Lambda-Ereignisnutzlast entweder Token für einen `TOKEN`-Genehmiger oder Anfrage für einen `REQUEST`-Genehmiger.
9. Abhängig von der Auswahl im vorherigen Schritt führen Sie einen der folgenden Schritte aus:

   a. Gehen Sie für die Option Token wie folgt vor:

   - Geben Sie unter Token-Quelle den Header-Namen ein, der das Genehmigungstoken enthält. Der API-Client muss einen Header mit diesem Namen enthalten, um das Autorisierungs-Token an den Lambda-Genehmiger zu senden.
   - Geben Sie optional für die Token-Validierung eine RegEx Anweisung ein. API Gateway führt eine erste Überprüfung des eingegebenen Token für diesen Ausdruck an und ruft nach der erfolgreichen Validierung den Genehmiger auf. Dies trägt dazu bei, die Anzahl der Aufrufe Ihrer API zu reduzieren.
   - Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn ein Caching der Richtlinie aktiviert

ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert. Wenn das Richtlinien-Caching aktiviert ist, wird der unter Token-Quelle angegebene Header-Name zum Cache-Schlüssel. Wenn in der Anfrage mehrere Werte an diesen Header übergeben werden, werden alle Werte zum Cache-Schlüssel, wobei die Reihenfolge beibehalten wird.

> **Note**
>
> Die TTL-Standardwert ist 300 Sekunden. Der Höchstwert ist 3.600 Sekunden; dieses Limit kann nicht erhöht werden.

b. Für die Optionen Request machen Sie Folgendes:

- Wählen Sie für Identitätsquellen-Typ einen Parametertyp aus. Die unterstützten Parametertypen sind `Header`, `Query string`, `Stage variable` und `Context`. Um weitere Identitätsquellen hinzuzufügen, wählen Sie Parameter hinzufügen.
- Um die vom Genehmiger generierte Genehmigungsrichtlinie zwischenzuspeichern, lassen Sie Genehmigungs-Caching aktiviert. Wenn ein Caching der Richtlinie aktiviert ist, können Sie den TTL-Wert ändern. Durch die Einstellung TTL = 0 wird das Richtlinien-Caching deaktiviert.

  API Gateway verwendet die angegebenen Identitätsquellen als Cachingschlüssel für den Anfrageautorisierer. Wenn das Caching aktiviert ist, ruft API Gateway die Lambda-Funktion des Genehmigers erst auf, nachdem erfolgreich überprüft wurde, dass alle angegebenen Identitätsquellen zur Laufzeit vorhanden sind. Wenn eine angegebene Identifikationsquelle fehlt, Null oder leer ist, gibt API Gateway eine `401 Unauthorized`-Antwort zurück, ohne die Lambda-Genehmigerfunktion aufzurufen.

  Wenn mehrere Identitätsquellen definiert sind, werden sie alle verwendet, um den Cache-Schlüssel des Genehmigers abzuleiten. Das Ändern von Teilen des Cache-Schlüssels bewirkt, dass der Genehmiger das im Cache befindliche Richtliniendokument verwirft und ein neues erstellt. Wenn in der Anfrage ein Header mit mehreren Werten übergeben wird, werden alle Werte Teil des Cache-Schlüssels, wobei die Reihenfolge beibehalten wird.
- Wenn das Caching deaktiviert ist, ist es nicht erforderlich, eine Identitätsquelle anzugeben.

> **Note**
>
> Zum Aktivieren der Zwischenspeicherung muss der Genehmiger eine Richtlinie zurückgeben, die für alle Methoden einer API anwendbar ist. Um methodenspezifische Richtlinien durchzusetzen, können Sie Genehmiger-Caching deaktivieren.

10. Wählen Sie Genehmiger erstellen aus.
11. Fügen Sie die `aws lambda add-permission` Befehlszeichenfolge, die Sie in einem vorherigen Schritt kopiert haben, in ein AWS CLI Fenster ein, das für Ihr zweites Konto konfiguriert ist. Ersetzen Sie `AUTHORIZER_ID` durch die ID Ihres Genehmigers. Dadurch erhält Ihr erstes Konto Zugriff auf die Lambda-Genehmigerfunktion Ihres zweiten Kontos.

Zugriff auf eine REST-API mit Amazon Cognito-Benutzerpools als Genehmiger steuern

Als Alternative zur Verwendung von IAM-Rollen und Richtlinien oder eines Lambda-Genehmigers (früher als benutzerdefinierter Genehmiger bekannt), können Sie einen Amazon Cognito-Benutzerpool verwenden, um den Zugriff auf Ihre API in Amazon API Gateway zu steuern.

Um einen Amazon Cognito-Benutzerpool mit Ihrer API zu verwenden, müssen Sie zunächst einen Genehmiger vom Typ `COGNITO_USER_POOLS` erstellen und dann eine API-Methode zur Verwendung dieses Genehmigers konfigurieren. Nach der Bereitstellung der API muss sich der Client zunächst am Benutzerpool anmelden, ein Identitäts- oder Zugriffstoken für den Benutzer abrufen und dann die API-Methode mit einem dieser Token, die in der Regel auf den `Authorization`-Header der Anfrage festgelegt sind, aufrufen. Der API-Aufruf ist nur erfolgreich, wenn der erforderliche Token übergeben und gültig ist. Andernfalls wird der Client nicht für den Aufruf autorisiert, da der Client über keine autorisierbaren Anmeldeinformationen verfügt.

Mit dem Identitätstoken werden API-Aufrufe auf Grundlage der Identitätsansprüche des angemeldeten Benutzers autorisiert. Mit dem Zugriffstoken werden API-Aufrufe auf Grundlage der benutzerdefinierten Bereiche von angegebenen zugriffsgeschützten Ressourcen autorisiert. Weitere Informationen finden Sie unter Verwenden von Token mit Benutzerpools und Ressourcenserver und benutzerdefinierte Bereiche.

Um einen Amazon Cognito-Benutzerpool für Ihre API zu erstellen und zu konfigurieren, führen Sie die folgenden Aufgaben aus:

- Verwenden Sie die Amazon Cognito Cognito-Konsole, das CLI/SDK oder die API, um einen Benutzerpool zu erstellen — oder verwenden Sie einen Pool, der einem anderen Konto gehört. AWS
- Verwenden Sie die API Gateway-Konsole, die CLI, das SDK oder die API, um einen API Gateway-Genehmiger mit dem ausgewählten Benutzerpool zu erstellen.
- Verwenden Sie die API Gateway-Konsole, die CLI, das SDK oder die API, um den Genehmiger für ausgewählte API-Methoden zu aktivieren.

Um alle API-Methoden mit einem Benutzerpool zu aktivieren, führen Ihre API-Clients die folgenden Aufgaben aus:

- Verwenden Sie die Amazon Cognito CLI, das SDK oder die API, um einen Benutzer beim gewählten Benutzerpool anzumelden und ein Identitäts- oder Zugriffstoken zu erhalten. Weitere Informationen zur Verwendung der SDKs finden Sie unter Codebeispiele für Amazon Cognito using AWS SDKs.
- Verwenden Sie ein kundenspezifisches Framework, um die bereitgestellte API Gateway-API aufzurufen und das entsprechende Token im `Authorization`-Header bereitzustellen.

Als API-Entwickler müssen Sie Ihren Client-Entwicklern die Benutzerpool-ID, eine Client-ID und die entsprechenden Clientschlüssel bereitstellen, die als Teil des Benutzerpools definiert werden.

> **Note**
>
> Verwenden Sie Amazon Cognito Federated Identities, damit sich ein Benutzer mit Anmeldeinformationen von Amazon Cognito anmelden und temporäre Anmeldeinformationen zur Verwendung mit den Berechtigungen einer IAM-Rolle erhalten kann. Legen Sie bei jeder HTTP-Methode für API-Ressourcenendpunkte den Autorisierungstyp, Kategorie `Method Execution` auf `AWS_IAM` fest.

In diesem Abschnitt wird beschrieben, wie Sie einen Benutzerpool erstellen, eine API Gateway-API in den Benutzerpool integrieren und eine API aufrufen, die im Benutzerpool integriert ist.

Themen

- Berechtigungen zur Erstellung von Amazon Cognito-Benutzerpool-Genehmigern für eine REST-API abrufen

- Amazon Cognito-Benutzerpool für eine REST-API erstellen
- REST-API mit einem Amazon Cognito-Benutzerpool integrieren
- Mit einem Amazon Cognito-Benutzerpool integrierte REST-API aufrufen
- Kontoübergreifenden Amazon Cognito-Genehmiger für eine REST-API über die API Gateway-Konsole konfigurieren
- Erstellen Sie einen Amazon Cognito Cognito-Authorizer für eine REST-API mit AWS CloudFormation

Berechtigungen zur Erstellung von Amazon Cognito-Benutzerpool-Genehmigern für eine REST-API abrufen

Um einen Genehmiger mit einem Amazon Cognito-Benutzerpool zu erstellen, müssen Sie über `Allow`-Berechtigungen zur Erstellung eines Genehmigers oder zur Aktualisierung eines Genehmigers mit dem ausgewählten Amazon Cognito-Benutzerpool verfügen. Das folgende IAM-Richtliniendokument zeigt ein Beispiel solcher Berechtigungen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST"
            ],
            "Resource": "arn:aws:apigateway:*::/restapis/*/authorizers",
            "Condition": {
                "ArnLike": {
                    "apigateway:CognitoUserPoolProviderArn": [
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_aD06NQmj0",
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-
east-1_xJ1MQtPEN"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:PATCH"
```

```
            ],
            "Resource": "arn:aws:apigateway:*::/restapis/*/authorizers/*",
            "Condition": {
                "ArnLike": {
                    "apigateway:CognitoUserPoolProviderArn": [
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_aD06NQmjO",
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-
east-1_xJ1MQtPEN"
                    ]
                }
            }
        }
    ]
}
```

Stellen Sie sicher, dass die Richtlinie einer IAM-Gruppe angefügt ist, der Sie angehören, oder einer IAM-Rolle, der Sie zugewiesen sind.

Im vorangegangenen Richtliniendokument wird die `apigateway:POST`-Aktion für das Erstellen eines neuen Genehmigers und die `apigateway:PATCH`-Aktion für das Aktualisieren eines vorhandenen Genehmigers verwendet. Sie können die Richtlinie auf eine bestimmte Region oder eine bestimmte API beschränken, indem Sie die ersten zwei Platzhalterzeichen (*) der `Resource`-Werte entsprechend überschreiben.

Die hier verwendeten `Condition`-Klauseln sollen die `Allowed`-Berechtigungen für die angegebenen Benutzerpools einschränken. Wenn eine `Condition`-Klausel vorhanden ist, wird der Zugriff auf alle Benutzerpools, die den Bedingungen nicht entsprechen, verweigert. Wenn eine Berechtigung über keine `Condition`-Klausel verfügt, wird der Zugriff auf alle Benutzerpools erlaubt.

Sie verfügen über folgende Optionen, um die `Condition`-Klausel festzulegen:

- Sie können den bedingten Ausdruck `ArnLike` oder `ArnEquals` festlegen, um die Erstellung oder Aktualisierung von `COGNITO_USER_POOLS`-Genehmigern nur mit den angegebenen Benutzerpools zu gestatten.
- Sie können den bedingten Ausdruck `ArnNotLike` oder `ArnNotEquals` festlegen, um die Erstellung oder Aktualisierung von `COGNITO_USER_POOLS`-Genehmigern mit allen nicht im Ausdruck angegebenen Benutzerpools zu gestatten.

- Sie können die `Condition`-Klausel weglassen, um das Erstellen oder Aktualisieren von `COGNITO_USER_POOLS`-Genehmigern mit allen Benutzerpools, von allen AWS -Konten und in jeder Region zu gestatten.

Weitere Informationen zu bedingten Amazon-Ressourcenname (ARN)-Ausdrücken finden Sie unter Bedingungsoperatoren für Amazon-Ressourcennamen (ARN). Wie im Beispiel gezeigt, ist `apigateway:CognitoUserPoolProviderArn` eine Liste von ARNs der `COGNITO_USER_POOLS`-Benutzerpools, die mit einem API Gateway-Genehmiger vom Typ `COGNITO_USER_POOLS` verwendet werden können.

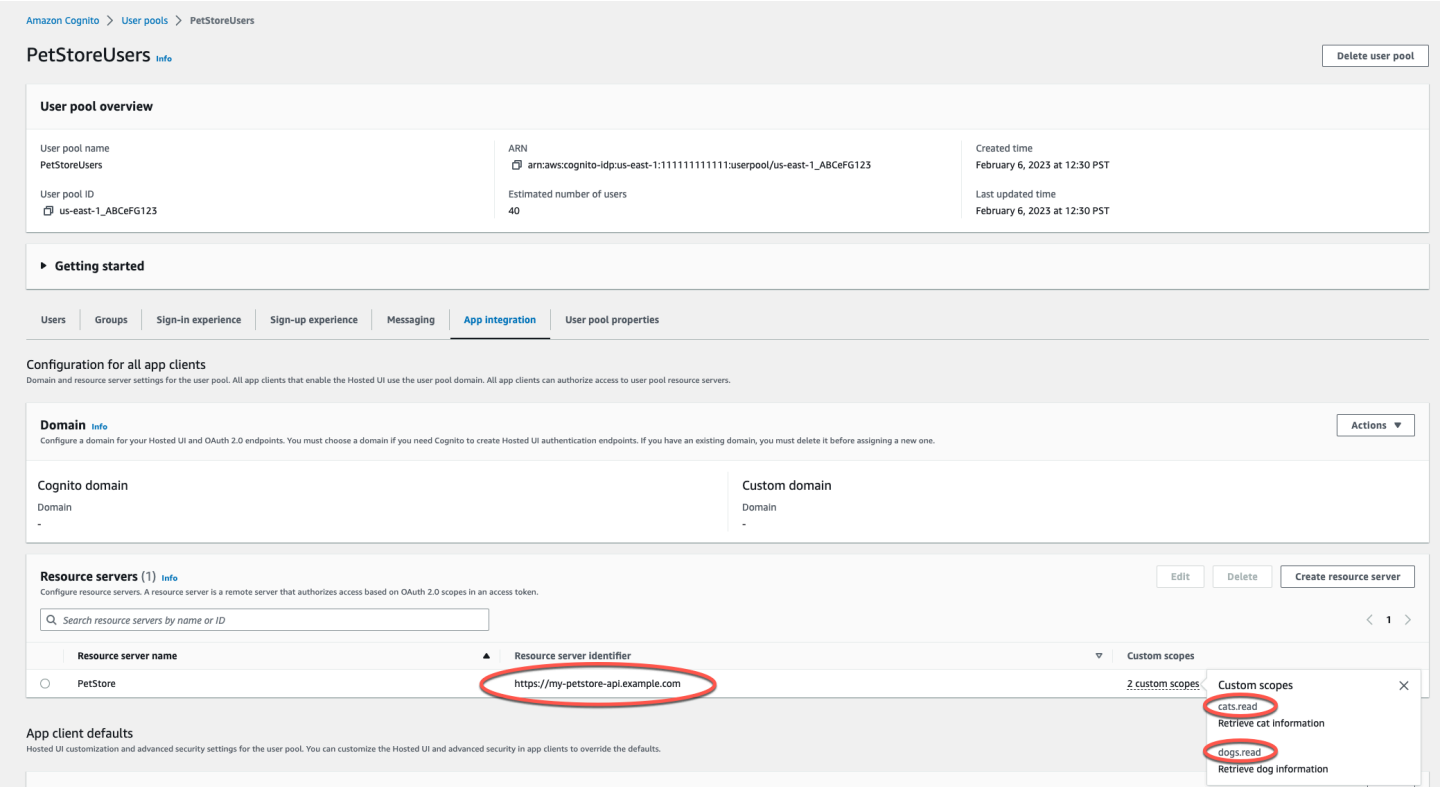
Amazon Cognito-Benutzerpool für eine REST-API erstellen

Bevor Sie Ihre API in einen Benutzerpool integrieren, müssen Sie den Benutzerpool in Amazon Cognito erstellen. Ihre Benutzerpoolkonfiguration muss alle Ressourcenkontingente für Amazon Cognito einhalten. Alle benutzerdefinierten Amazon-Cognito-Variablen wie Gruppen, Benutzer und Rollen dürfen nur alphanumerische Zeichen verwenden. Eine Anleitung zum Erstellen eines Benutzerpools finden Sie unter Tutorial: Erstellen eines Benutzerpools im Amazon-Cognito-Entwicklerhandbuch.

Beachten Sie die Benutzerpool-ID, Client-ID und jeden Clientschlüssel. Der Client muss diese an Amazon Cognito übermitteln, um Benutzer für den Benutzerpool zu registrieren, die Anmeldung am Benutzerpool auszuführen und der Identitäts- oder Zugriffstoken abzurufen, das in die Anfragen für Aufrufe von API-Methoden eingebunden werden muss, die für den Benutzerpool konfiguriert sind. Außerdem müssen Sie den Namen des Benutzerpools angeben, wenn Sie den Benutzerpool als Genehmiger in API Gateway konfigurieren, wie nachfolgend beschrieben.

Wenn Sie API-Methodenaufrufe mit Zugrifftokens autorisieren, stellen Sie bei der Konfiguration der App-Integration beim Benutzerpool sicher, dass die für einen bestimmten Ressourcenserver gewünschten benutzerdefinierten Bereiche eingerichtet werden. Weitere Informationen über die Verwendung von Tokens mit Amazon Cognito-Benutzerpools finden Sie unter Tokens mit Benutzerpools verwenden. Weitere Informationen zu Ressourcenservern finden Sie unter Definieren von Ressourcenservern für eigene Benutzerpools.

Beachten Sie die konfigurierten Ressourcenserver-IDs und benutzerdefinierten Bereichsnamen. Diese sind für die Erstellung der vollständigen Namen der Zugriffsbereiche für OAuth Scopes erforderlich, die vom `COGNITO_USER_POOLS`-Genehmiger verwendet werden.



REST-API mit einem Amazon Cognito-Benutzerpool integrieren

Nach der Erstellung eines Amazon Cognito-Benutzerpools müssen Sie in API Gateway einen `COGNITO_USER_POOLS`-Genehmiger erstellen, der den Benutzerpool verwendet. Im folgenden Verfahren wird gezeigt, wie Sie dies mithilfe der API-Gateway-Konsole tun.

> **Note**
>
> Sie können mit der `CreateAuthorizer`-Aktion einen `COGNITO_USER_POOLS`-Autorisierer erstellen, der mehrere Benutzerpools verwendet. Sie können bis zu 1 000 Benutzerpools für einen `COGNITO_USER_POOLS`-Autorisierer verwenden. Dieses Limit kann nicht erhöht werden.

> **Important**
>
> Nachdem Sie eines der folgenden Verfahren durchgeführt haben, müssen Sie Ihre API bereitstellen oder erneut bereitstellen, um die Änderungen zu übernehmen. Weitere

Informationen zum Bereitstellen Ihrer API finden Sie unter Bereitstellen einer REST-API in Amazon API Gateway.

So erstellen Sie einen **`COGNITO_USER_POOLS`**-Genehmiger unter Verwendung der API Gateway-Konsole:

1. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API in API Gateway aus.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Wählen Sie Genehmiger erstellen aus.
4. Zur Konfiguration des neuen Genehmigers für die Verwendung eines Benutzerpools führen Sie die folgenden Schritte aus:
   a. Geben Sie unter Name des Genehmigers einen Namen ein.
   b. Wählen Sie als Genehmiger-Typ Cognito aus.
   c. Wählen Sie für den Cognito-Benutzerpool den Ort aus, AWS-Region an dem Sie Ihren Amazon Cognito erstellt haben, und wählen Sie einen verfügbaren Benutzerpool aus.

      Sie können eine Stufenvariable verwenden, um Ihren Benutzerpool zu definieren. Verwenden Sie das folgende Format für Ihren Benutzerpool:`arn:aws:cognito-idp:`*`us-east-2`*`:`*`111122223333`*`:userpool/${stageVariables.`*`MyUserPool`*`}`.
   d. Geben Sie für Token-Quelle als Header-Name **`Authorization`** ein, um das Identitäts- oder Zugriffstoken zu übergeben, das von Amazon Cognito bei erfolgreicher Anmeldung eines Benutzers zurückgegeben wird.
   e. Optional können Sie einen regulären Ausdruck im Feld Tokenvalidierung eingeben, um das `aud`-Feld (Zielgruppe) des Identitätstokens auszuwerten, bevor die Anfrage mit Amazon Cognito genehmigt wird. Beachten Sie, dass diese Validierung bei Verwendung eines Zugriffstoken die Anforderung zurückweist, da das Zugriffstoken das `aud`-Feld nicht enthält.
   f. Wählen Sie Genehmiger erstellen aus.
5. Nach der Erstellung des `COGNITO_USER_POOLS`-Genehmigers können Sie einen optionalen Testaufruf durchführen, indem Sie ein vom Benutzerpool bereitgestelltes Identitätstoken übergeben. Sie können dieses Identitätstoken erhalten, indem Sie das Amazon Cognito Identity-SDK aufrufen, um die Benutzeranmeldung durchzuführen. Sie können auch die `InitiateAuth`-Aktion verwenden. Wenn Sie keine Genehmigungsbereiche konfigurieren, behandelt API Gateway das bereitgestellte Token als Identitätstoken.

Das vorhergehende Verfahren erstellt einen `COGNITO_USER_POOLS`-Genehmiger, der den neu erstellten Amazon Cognito-Benutzerpool verwendet. Je nach der Aktivierungsweise des Genehmigers mit einer API-Methode können Sie entweder ein Identitäts- oder Zugriffstoken verwenden, das vom integrierten Benutzerpool bereitgestellt wird.

So konfigurieren Sie einen **`COGNITO_USER_POOLS`**-Genehmiger mit Methoden

1. Wählen Sie Resources aus. Wählen Sie eine neue oder eine vorhandene Methode aus. Falls erforderlich, erstellen Sie eine Ressource.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Wählen Sie für Genehmiger aus dem Dropdown-Menü die Amazon-Cognito-Benutzerpool-Genehmiger, die Sie soeben erstellt haben.
4. Zur Verwendung eines Identitätstokens führen Sie die folgenden Schritte aus:

   a. Lassen Sie das Feld Genehmigungsbereiche leer.

   b. Fügen Sie bei Bedarf unter Integrationsanforderung den Ausdruck `$context.authorizer.claims['`*`property-name`*`']` oder `$context.authorizer.claims.`*`property-name`* in einer Text-Mapping-Vorlage hinzu, um die angegebene Eigenschaft für Identitätsansprüche vom Benutzerpool an das Backend zu übergeben. Bei einfachen Eigenschaftsnamen (wie z. B. `sub` oder `custom-sub`) sind beide Notationen gleich. Bei komplexen Eigenschaftsnamen (wie z. B. `custom:role`) darf keine punktierte Schreibweise verwendet werden. Beispielsweise werden die Standardfelder `sub` und `email` des Anspruchs mit den folgenden Mapping-Ausdrücken an das Backend übergeben:

   ```
   {
    "context" : {
     "sub" : "$context.authorizer.claims.sub",
     "email" : "$context.authorizer.claims.email"
    }
   }
   ```

   Wenn Sie bei der Konfiguration des Benutzerpools ein Feld für einen benutzerdefinierten Anspruch deklariert haben, können Sie auf die gleiche Art auf die benutzerdefinierten Felder zugreifen. Im folgenden Beispiel wird das benutzerdefinierte Feld `role` eines Anspruchs abgerufen:

```
{
 "context" : {
  "role" : "$context.authorizer.claims.role"
    }
}
```

Wurde das Feld für den benutzerdefinierten Anspruch mit `custom:role` deklariert, verwenden Sie folgendes Beispiel für den Abruf der Anspruchseigenschaft:

```
{
 "context" : {
  "role" : "$context.authorizer.claims['custom:role']"
    }
}
```

5. Zur Verwendung eines Zugriffstokens führen Sie die folgenden Schritte aus:

   a. Geben Sie für Genehmigungsbereiche einen oder mehrere vollständige Namen eines Bereichs ein, der bei der Erstellung des Amazon-Cognito-Benutzerpools konfiguriert wurde. Im Beispiel, das in Amazon Cognito-Benutzerpool für eine REST-API erstellen gezeigt wurde, ist einer der Bereiche beispielsweise `https://my-petstore-api.example.com/cats.read`.

   Bei Laufzeit ist der Methodenaufruf dann erfolgreich, wenn ein Bereich, der in diesem Schritt in der Methode angegeben wird, einem Bereich entspricht, der im eingehenden Token beantragt wird. Andernfalls schlägt der Aufruf mit einer `401 Unauthorized`-Antwort fehl.

   b. Wählen Sie Speichern.

6. Wiederholen Sie diese Schritte für weitere Methoden Ihrer Wahl.

Wenn beim `COGNITO_USER_POOLS`-Genehmiger die Option OAuth Scopes nicht angegeben wird, behandelt Amazon API Gateway den angegebenen Token als Identitätstoken und bestätigt die beantragte Identität mit der aus dem Benutzerpool. Andernfalls behandelt API Gateway den angegebenen Token als Zugriffstoken und prüft die im Token beanspruchten Zugriffsbereiche gegen die in der Methode deklarierten Autorisierungsbereiche.

Statt die API Gateway-Konsole zu verwenden, können Sie auch einen Amazon Cognito-Benutzerpool für eine Methode aktivieren, indem Sie eine OpenAPI-Definitionsdatei angeben und die API-Definition in API Gateway importieren.

So importieren Sie einen COGNITO_USER_POOLS-Genehmiger mit einer Swagger-Definitionsdatei

1. Erstellen (oder exportieren) Sie eine Swagger-Definitionsdatei für Ihre API.
2. Geben Sie die `COGNITO_USER_POOLS` -Genehmiger (`MyUserPool`) -Definition als Teil der JSON- `securitySchemes` Abschnitt in OpenAPI 3.0 oder den `securityDefinitions` Abschnitt im Open API-2.0 wie folgt an:

   OpenAPI 3.0

   ```
   "securitySchemes": {
     "MyUserPool": {
       "type": "apiKey",
       "name": "Authorization",
       "in": "header",
       "x-amazon-apigateway-authtype": "cognito_user_pools",
       "x-amazon-apigateway-authorizer": {
         "type": "cognito_user_pools",
         "providerARNs": [
           "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
         ]
       }
     }
   ```

   OpenAPI 2.0

   ```
   "securityDefinitions": {
     "MyUserPool": {
       "type": "apiKey",
       "name": "Authorization",
       "in": "header",
       "x-amazon-apigateway-authtype": "cognito_user_pools",
       "x-amazon-apigateway-authorizer": {
         "type": "cognito_user_pools",
         "providerARNs": [
           "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
         ]
       }
     }
   ```

3. Wenn Sie den Identitätstoken für die Methodenautorisierung verwenden möchten, fügen Sie der `{ "MyUserPool": [] }`-Definition der Methode `security` hinzu, wie in der folgenden GET-Methode auf der Root-Ressource dargestellt.

```
"paths": {
  "/": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "text/html"
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        }
      },
      "security": [
        {
          "MyUserPool": []
        }
      ],
      "x-amazon-apigateway-integration": {
        "type": "mock",
        "responses": {
          "default": {
            "statusCode": "200",
            "responseParameters": {
              "method.response.header.Content-Type": "'text/html'"
            },
          }
        },
        "requestTemplates": {
          "application/json": "{\"statusCode\": 200}"
        },
        "passthroughBehavior": "when_no_match"
      }
    },
    ...
 }
```

4. Um den Zugriffstoken für die Methodenautorisierung zu verwenden, ändern Sie die obige Sicherheitsdefinition in `{ "MyUserPool": [resource-server/scope, ...] }`:

```
 "paths": {
   "/": {
     "get": {
       "consumes": [
         "application/json"
       ],
       "produces": [
         "text/html"
       ],
       "responses": {
         "200": {
           "description": "200 response",
           "headers": {
             "Content-Type": {
               "type": "string"
             }
           }
         }
       },
       "security": [
         {
           "MyUserPool": ["https://my-petstore-api.example.com/cats.read",
"http://my.resource.com/file.read"]
         }
       ],
       "x-amazon-apigateway-integration": {
         "type": "mock",
         "responses": {
           "default": {
             "statusCode": "200",
             "responseParameters": {
               "method.response.header.Content-Type": "'text/html'"
             },
           }
         },
         "requestTemplates": {
           "application/json": "{\"statusCode\": 200}"
         },
         "passthroughBehavior": "when_no_match"
       }
```

```
    },
    ...
}
```

5. Bei Bedarf können Sie mithilfe der geeigneten Swagger-Definitionen oder -Erweiterungen weitere API-Konfigurationseinstellungen festlegen. Weitere Informationen finden Sie unter API Gateway-Erweiterungen für OpenAPI arbeiten.

Mit einem Amazon Cognito-Benutzerpool integrierte REST-API aufrufen

Vom Client müssen die folgenden Aufgaben ausführt werden, um eine Methode mit konfiguriertem Benutzerpool-Genehmiger aufzurufen:

- Der Benutzer muss für den Benutzerpool registriert sein.
- Der Benutzer muss sich am Benutzerpool anmelden.
- Rufen Sie ein Identitäts- oder Zugriffs-Token des angemeldeten Benutzers vom Benutzerpool ab.
- Das Token muss in den `Authorization`-Header (oder einen anderen Header, der beim Erstellen des Genehmigers festgelegt wurde) eingebunden werden.

Sie können AWS Amplify verwenden, um diese Aufgaben auszuführen. Weitere Informationen finden Sie unter Integration von Amazon Cognito mit Web- und Mobile-Apps .

- Informationen zu Android finden Sie unter Erste Schritte mit Amplify für Android.
- Informationen zur Verwendung von iOS finden Sie unter Erste Schritte mit Amplify für iOS.
- Informationen zur Verwendung JavaScript finden Sie unter Erste Schritte mit Amplify für Javascript.

Kontoübergreifenden Amazon Cognito-Genehmiger für eine REST-API über die API Gateway-Konsole konfigurieren

Sie können jetzt auch einen Amazon Cognito Cognito-Benutzerpool von einem anderen AWS Konto aus als API-Autorisierer verwenden. Der Amazon Cognito-Benutzerpool kann Bearer-Token-Authentifizierungsstrategien wie OAuth oder SAML verwenden. Dadurch ist es einfach, einen zentralen Genehmiger eines Amazon Cognito-Benutzerpools über mehrere API Gateway-APIs hinweg zentral zu verwalten und gemeinsam zu nutzen.

In diesem Abschnitt zeigen wir, wie ein kontenübergreifender Amazon Cognito-Benutzerpool mit Hilfe der Amazon API Gateway-Konsole konfiguriert wird.

Bei diesen Anweisungen wird davon ausgegangen, dass Sie bereits eine API Gateway Gateway-API in einem AWS Konto und einen Amazon Cognito Cognito-Benutzerpool in einem anderen Konto haben.

Kontoübergreifenden Amazon Cognito-Genehmiger mit der API Gateway-Konsole konfigurieren

Melden Sie sich in der Amazon-API-Gateway-Konsole bei dem Konto an, in dem Ihre API enthalten ist, und gehen Sie wie folgt vor:

1. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API in API Gateway aus.
2. Wählen Sie im Hauptnavigationsbereich Genehmiger.
3. Wählen Sie Genehmiger erstellen aus.
4. Zur Konfiguration des neuen Genehmigers für die Verwendung eines Benutzerpools führen Sie die folgenden Schritte aus:

   a. Geben Sie unter Name des Genehmigers einen Namen ein.

   b. Wählen Sie als Genehmiger-Typ Cognito aus.

   c. Geben Sie den vollständigen ARN für den Benutzerpool in Ihrem zweiten Konto in das Feld Cognito-Benutzerpool ein.

   > **Note**
   >
   > In der Amazon Cognito-Konsole finden Sie den ARN für eigene Benutzerpools im Feld Pool ARN des Bereichs General Settings (Allgemeine Einstellungen).

   d. Geben Sie für Token-Quelle als Header-Name **`Authorization`** ein, um das Identitäts- oder Zugriffstoken zu übergeben, das von Amazon Cognito bei erfolgreicher Anmeldung eines Benutzers zurückgegeben wird.

   e. Optional können Sie einen regulären Ausdruck im Feld Tokenvalidierung eingeben, um das `aud`-Feld (Zielgruppe) des Identitätstokens auszuwerten, bevor die Anfrage mit Amazon Cognito genehmigt wird. Beachten Sie, dass diese Validierung bei Verwendung eines Zugriffstoken die Anforderung zurückweist, da das Zugriffstoken das `aud`-Feld nicht enthält.

   f. Wählen Sie Genehmiger erstellen aus.

Erstellen Sie einen Amazon Cognito Cognito-Authorizer für eine REST-API mit AWS CloudFormation

Sie können AWS CloudFormation damit einen Amazon Cognito Cognito-Benutzerpool und einen Amazon Cognito Cognito-Autorisierer erstellen. Die AWS CloudFormation Beispielvorlage macht Folgendes:

- Erstellen eines Amazon-Cognito-Benutzerpools. Der Client muss den Benutzer zunächst beim Benutzerpool anmelden und ein Identitäts- oder Zugriffstoken abrufen. Wenn Sie API-Methodenaufrufe mit Zugrifftokens autorisieren, stellen Sie bei der Konfiguration der App-Integration beim Benutzerpool sicher, dass die für einen bestimmten Ressourcenserver gewünschten benutzerdefinierten Bereiche eingerichtet werden.
- Erzeugt eine API-Gateway-API mit einer `GET`-Methode
- Erstellt einen Amazon-Cognito-Genehmiger, der den `Authorization`-Header als Token-Quelle verwendet.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      AccountRecoverySetting:
        RecoveryMechanisms:
          - Name: verified_phone_number
            Priority: 1
          - Name: verified_email
            Priority: 2
      AdminCreateUserConfig:
        AllowAdminCreateUserOnly: true
      EmailVerificationMessage: The verification code to your new account is {####}
      EmailVerificationSubject: Verify your new account
      SmsVerificationMessage: The verification code to your new account is {####}
      VerificationMessageTemplate:
        DefaultEmailOption: CONFIRM_WITH_CODE
        EmailMessage: The verification code to your new account is {####}
        EmailSubject: Verify your new account
        SmsMessage: The verification code to your new account is {####}
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  CogAuthorizer:
    Type: AWS::ApiGateway::Authorizer
```

```
    Properties:
      Name: CognitoAuthorizer
      RestApiId:
        Ref: Api
      Type: COGNITO_USER_POOLS
      IdentitySource: method.request.header.Authorization
      ProviderARNs:
        - Fn::GetAtt:
            - UserPool
            - Arn
  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyCogAuthApi
  ApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
        Ref: Api
    DependsOn:
      - CogAuthorizer
      - ApiGET
  ApiDeploymentStageprod:
    Type: AWS::ApiGateway::Stage
    Properties:
      RestApiId:
        Ref: Api
      DeploymentId:
        Ref: ApiDeployment
      StageName: prod
  ApiGET:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      ResourceId:
        Fn::GetAtt:
          - Api
          - RootResourceId
      RestApiId:
        Ref: Api
      AuthorizationType: COGNITO_USER_POOLS
      AuthorizerId:
        Ref: CogAuthorizer
      Integration:
```

```
        IntegrationHttpMethod: GET
        Type: HTTP_PROXY
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
Outputs:
  ApiEndpoint:
    Value:
      Fn::Join:
        - ""
        - - https://
          - Ref: Api
          - .execute-api.
          - Ref: AWS::Region
          - "."
          - Ref: AWS::URLSuffix
          - /
          - Ref: ApiDeploymentStageprod
          - /
```

REST-API-Integrationen einrichten

Nach dem Einrichten einer API-Methode müssen Sie diese in einen Endpunkt im Backend integrieren. Ein Backend-Endpunkt wird auch als Integrationsendpunkt bezeichnet und kann eine Lambda-Funktion, eine HTTP-Webseite oder eine AWS Serviceaktion sein.

Wie bei der API-Methode auch beinhaltet die API-Integration eine Integrationsanforderung eine Integrationsantwort. Die Integrationsanforderung kapselt eine HTTP-Anforderung vom Backend. Diese kann, muss sich aber nicht von der vom Client gesendeten Methodenanforderung unterscheiden. Die Integrationsantwort ist eine HTTP-Antwort, die die Ausgabe des Backends kapselt.

Das Einrichten einer Integrationsanforderung umfasst die folgenden Schritte: Konfiguration, wie vom Client gesendete Methodenanforderungen an das Backend übergeben werden, Konfiguration, wie die Anfragedaten gegebenenfalls in die Integrationsanforderungsdaten umgewandelt werden, Festlegen der aufzurufenden Lambda-Funktion, Festlegen des HTTP-Servers, an den die eingehende Anfrage weitergeleitet wird oder Festlegen, welche AWS -Service-Aktion aufgerufen wird.

Das Einrichten einer Integrationsantwort (ausschließlich auf Nicht-Proxy-Integrationen anwendbar) umfasst die folgenden Schritte: Konfiguration, wie das vom Backend zurückgegebene Ergebnis an eine Methodenantwort eines gegebenen Statuscodes übergeben wird, Konfiguration, wie die festgelegten Parameter der Integrationsantwort in vorkonfigurierte Parameter der Methodenantwort

umgewandelt werden und Konfiguration, wie der Inhalt der Integrationsantwort gemäß den angegebenen Mapping-Vorlagen dem Inhalt der Methodenantwort zugeordnet wird.

Programmgesteuert wird eine Integrationsanforderung durch die Ressource `Integration` und eine Integrationsantwort durch die Ressource `IntegrationResponse` von Amazon API Gateway gekapselt.

Um eine Integrationsanforderung einzurichten, erstellen Sie eine `Integration`-Ressource und verwenden diese zum Konfigurieren der Integrations-Endpunkt-URL. Dann richten Sie die IAM-Berechtigungen für den Zugriff auf das Backend ein und legen die Mappings für die Umwandlung der eingehenden Anfragedaten vor der Übergabe an das Backend fest. Um eine Integrationsantwort für eine Nicht-Proxy-Integration einzurichten, erstellen Sie eine `IntegrationResponse`-Ressource und verwenden diese zum Festlegen der Zielmethodenantwort. Dann konfigurieren Sie, wie die Backend-Ausgabe der Methodenausgabe zugeordnet wird.

Themen

Integrationsanfrage in API Gateway einrichten

Führen Sie die folgenden erforderlichen und optionalen Schritte aus, um eine Integrationsanforderung einzurichten:

1. Wählen Sie einen Integrationstyp aus, um festzulegen, wie Methodenanforderungsdaten an das Backend übergeben werden.
2. Geben Sie für alle Integrationen außerdem dem Integrationstyp `MOCK` eine HTTP-Methode und die URI des gewählten Integrationsendpunkts an.
3. Für Integrationen mit Lambda-Funktionen und anderen AWS Serviceaktionen richten Sie eine IAM-Rolle mit den erforderlichen Berechtigungen ein, damit API Gateway das Backend in Ihrem Namen aufruft.

4. Richten Sie für Nicht-Proxy-Integrationen die erforderlichen Parameter-Mappings ein, um vordefinierte Methodenanforderungs-Parameter den entsprechenden Integrationsanforderungs-Parametern zuzuordnen.
5. Richten Sie für Nicht-Proxy-Integrationen die erforderlichen Inhalts-Mappings ein, um den Inhalt der eingehenden Methodenanforderung eines bestimmten Inhaltstyps gemäß der angegebenen Mapping-Vorlage zuzuordnen.
6. Geben Sie für Nicht-Proxy-Integrationen die Bedingung an, unter der die eingehenden Methodenanforderungsdaten unverändert an das Backend übergeben werden.
7. Legen Sie optional fest, wie Typumwandlungen für binäre Nutzlasten verarbeitet werden sollen.
8. Deklarieren Sie optional einen Cache-Namespace-Namen und Cache-Schlüsselparameter, um API-Caching zu aktivieren.

Zur Durchführung dieser Aufgaben müssen Sie eine API Gateway-Integrations-Ressource erstellen und entsprechende Eigenschaftswerte festlegen. Sie können dazu die API Gateway Gateway-Konsole, AWS CLI Befehle, ein AWS SDK oder die API Gateway Gateway-REST-API verwenden.

Themen

- Grundlegende Aufgaben einer API-Integrationsanforderung
- Auswählen eines API Gateway-API-Integration
- Einrichten der Proxy-Integration mit einer Proxy-Ressource
- API-Integrationsanfrage über die API Gateway-Konsole einrichten

Grundlegende Aufgaben einer API-Integrationsanforderung

Eine Integrationsanforderung ist eine HTTP-Anfrage, die API Gateway an das Backend sendet. Dabei werden die vom Client gesendeten Anfragedaten übergeben und gegebenenfalls umgewandelt. Die HTTP-Methode (oder Verb) und URI der Integrationsanforderung werden vom Backend, also dem Integrationsendpunkt, vorgegeben. Sie können, müssen aber nicht identisch mit der HTTP-Methode und URI der Methodenanforderung übereinstimmen.

Wenn eine Lambda-Funktion eine Datei zurückgibt, die von Amazon S3 abgerufen wurde, können Sie diesen Vorgang beispielsweise intuitiv als `GET`-Methodenanfrage für den Client bereitstellen, obwohl die entsprechende Integrationsanfrage festlegt, dass eine `POST`-Anfrage zum Aufruf der Lambda-Funktion verwendet wird. Bei einem HTTP-Endpunkt ist es wahrscheinlich, dass für die Methodenanforderung und die entsprechende Integrationsanforderung dasselbe HTTP-Verb

verwendet wird. Dies ist jedoch nicht erforderlich. Sie können die folgende Methodenanforderung integrieren:

```
GET /{var}?query=value
Host: api.domain.net
```

Mit der folgenden Integrationsanforderung:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
   path: "{var}'s value",
   type: "value"
}
```

Als API-Entwickler können Sie beliebige HTTP-Verben und URIs für eine Methodenanforderung verwenden, die Ihre Anforderungen erfüllt. Dabei müssen jedoch die Anforderungen des Integrationsendpunkts erfüllt werden. Wenn die Methodenanforderungsdaten sich von den Daten der Integrationsanforderung unterscheiden, können Sie den Unterschied dadurch ausgleichen, dass Sie das Mapping der Methodenanforderungsdaten zu den Integrationsanforderungsdaten angeben.

In den vorherigen Beispielen werden die Werte für die Pfadvariable (`{var}`) und den Abfrageparameter (`query`) der `GET`-Methodenanforderung in Werte der Nutzlasteigenschaften der Integrationsanforderung `path` und `type` zugeordnet. Auch der Header und Inhalt der Anforderungsdaten können zugeordnet werden. Diese werden in Mappings von Anfrage- und Antwortdaten über die API Gateway-Konsole einrichten beschrieben.

Beim Einrichten der HTTP- oder HTTP-Proxy-Integrationsanforderung weisen Sie die URL des Backend-HTTP-Endpunkts als URI-Wert der Integrationsanforderung zu. In der PetStore API hat die Methodenanforderung zum Abrufen einer Seite mit Haustieren beispielsweise den folgenden URI für die Integrationsanforderung:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Beim Einrichten der Lambda- oder Lambda-Proxy-Integration weisen Sie den Amazon-Ressourcennamen (ARN) für den Aufruf der Lambda-Funktion als URI-Wert der Integrationsanforderung zu. Der ARN hat das folgende Format:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

Der Teil nach `arn:aws:apigateway:`*api-region*`:lambda:path/` (`/2015-03-31/functions/arn:aws:lambda:`*lambda-region*`:`*account-id*`:function:`*lambda-function-name*`/invocations`) ist der REST-API-URI-Pfad der Lambda-Aufruf-Aktion. Wenn Sie die API Gateway-Konsole zum Einrichten der Lambda-Integration verwenden, erstellt API Gateway den ARN und weist ihn der Integrations-URI zu, nachdem Sie aufgefordert wurden, den *lambda-function-name* aus einer Region auszuwählen.

Wenn die Integrationsanforderung mit einer anderen AWS Dienstaktion eingerichtet wird, ist der Integrationsanforderungs-URI ebenfalls ein ARN, ähnlich der Integration mit der `Invoke` Lambda-Aktion. Für die Integration mit der GetBucketAktion von Amazon S3 ist die Integrationsanforderungs-URI beispielsweise ein ARN mit dem folgenden Format:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

Die Integrationsanforderungs-URI entspricht der Pfadkonvention zur Festlegung der Aktion, wobei *{bucket}* der Platzhalter für einen Bucket-Namen ist. Alternativ kann auf eine AWS Serviceaktion anhand ihres Namens verwiesen werden. Mit dem Aktionsnamen sieht die Integrationsanforderungs-URI für die `GetBucket`-Aktion für Amazon S3 wie folgt aus:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

In der aktionsbasierten Integrationsanforderungs-URI muss der Bucket-Name (*{bucket}*) im Inhalt der Integrationsanforderung (`{ Bucket: "`*{bucket}*`" }`) im Eingabeformat der `GetBucket`-Aktion festgelegt werden.

Für AWS Integrationen müssen Sie auch Anmeldeinformationen konfigurieren, damit API Gateway die integrierten Aktionen aufrufen kann. Sie können eine neue IAM-Rolle erstellen oder eine vorhandene IAM-Rolle auswählen, mit der API Gateway die Aktion aufruft, und dann die Rolle mit ihrem ARN angeben. Hier ein Beispiel für diesen ARN:

```
arn:aws:iam::account-id:role/iam-role-name
```

Die IAM-Rolle muss eine Richtlinie enthalten, in der die Berechtigung für die auszuführende Aktion gewährt wird. Außerdem muss API Gateway (in der Vertrauensbeziehung der Rolle) als vertrauenswürdige Entität deklariert sein, um die Rolle übernehmen zu können. Solche Berechtigungen können für die Aktion selbst gewährt werden. Sie werden als ressourcenbasierte Berechtigungen bezeichnet. Für die Lambda-Integration können Sie die Lambda-Aktion addPermission aufrufen, um die ressourcenbasierten Berechtigungen festzulegen, und dann `credentials` in der API Gateway-Integrationsanforderung auf Null festlegen.

Wir haben die Grundlagen der Integrationseinrichtung besprochen. Die erweiterten Einstellungen umfassen die Möglichkeit des Mapping der Methodenanforderungsdaten zu den Integrationsanforderungsdaten. Nachdem wir die grundlegende Einrichtung für eine Integrationsantwort behandelt haben, decken wir fortgeschrittene Themen in Mappings von Anfrage- und Antwortdaten über die API Gateway-Konsole einrichten ab, wo auch die Übergabe von Nutzlasten und der Umgang mit Verschlüsselungen behandelt werden.

Auswählen eines API Gateway-API-Integration

Die Wahl des API-Integrationstyps erfolgt gemäß dem verwendeten Typs des Integrationsendpunkts und der Übergabemethode von Daten an und von dem Integrationsendpunkt. Für eine Lambda-Funktion können Sie die Lambda-Proxy-Integration oder die benutzerdefinierte Lambda-Integration verwenden. Für einen HTTP-Endpunkt können Sie die HTTP-Proxy-Integration oder die benutzerdefinierte HTTP-Integration verwenden. Für eine AWS Dienstaktion steht Ihnen nur die AWS Integration des Typs ohne Proxy zur Verfügung. API Gateway unterstützt auch die Mock-Integration, bei der API Gateway als Integrationsendpunkt dient, um auf eine Methodenanfragen zu antworten.

Die benutzerdefinierte Lambda-Integration ist ein Sonderfall der AWS Integration, bei dem der Integrationsendpunkt der funktionsaufrufenden Aktion des Lambda-Service entspricht.

Programmgesteuert wählen Sie einen Integrationstyp durch Festlegen der Eigenschaft `type` für die Ressource `Integration` aus. Für die Lambda-Proxy-Integration ist der Wert `AWS_PROXY`. Für die benutzerdefinierte Lambda-Integration und alle anderen AWS -Integrationen lautet er `AWS`. Für die HTTP-Proxy-Integration und die HTTP-Integration ist der Wert `HTTP_PROXY` bzw. `HTTP`. Für die Pseudo-Integration ist der `type`-Wert `MOCK`.

Die Lambda-Proxy-Integration unterstützt eine vereinfachte Integrationseinrichtung mit einer einzigen Lambda-Funktion. Die Einrichtung ist einfach und kann um das Backend erweitert werden, ohne die vorhandene Einrichtung zu beeinträchtigen. Aus diesen Gründen wird sie für die Integration mit einer Lambda-Funktion dringend empfohlen.

Mit der benutzerdefinierten Lambda-Integration können Sie dagegen konfigurierte Mapping-Vorlagen für verschiedene Integrationsendpunkte mit vergleichbaren Anfragen für die Ein- und Ausgabedatenformate wiederverwenden. Die Einrichtung ist aufwändiger und wird für erweiterte Anwendungsszenarios empfohlen.

Ebenso ist die Integrationseinrichtung für die HTTP-Proxy-Integration optimiert und kann um das Backend erweitert werden, ohne die bestehende Einrichtung zu beeinträchtigen. Die benutzerdefinierte HTTP-Integration ist aufwändiger einzurichten, kann jedoch in Form von konfigurierten Mapping-Vorlagen für andere Integrationsendpunkte wiederverwendet werden.

In der folgenden Liste sind die unterstützten Integrationstypen zusammengefasst:

- `AWS`: Mit diesem Integrationstyp kann eine API AWS -Service-Aktionen festlegen. Bei der `AWS`-Integration müssen Sie sowohl die Integrationsanforderung als auch die Integrationsantwort konfigurieren und die erforderlichen Daten-Mappings von der Methodenanforderung zur Integrationsanforderung sowie von der Integrationsantwort zur Methodenantwort einrichten.
- `AWS_PROXY`: Mit diesem Integrationstyp lassen sich API-Methoden in die Lambda-Funktionsaufrufaktion integrieren. Die Einrichtung ist flexibel, vielseitig und optimiert. Die Integration stützt sich auf direkte Interaktionen zwischen dem Client und der integrierten Lambda-Funktion.

  Bei diesem, auch als Lambda-Proxy-Integration bezeichneten Integrationstyp, legen Sie weder die Integrationsanforderung noch die Integrationsantwort fest. API Gateway übergibt die eingehende Anfrage vom Client als Eingabe an die Backend-Lambda-Funktion. Die integrierte Lambda-Funktion übernimmt die Eingabe in diesem Format und analysiert die Eingabe aus allen verfügbaren Quellen, darunter Anfrage-Header, URL-Pfadvariablen, Abfragezeichenfolge-Parameter und anwendbarer Inhalt. Die Funktion gibt das Ergebnis in diesem Ausgabeformat zurück.

  Dies ist der bevorzugte Integrationstyp zum Aufrufen einer Lambda-Funktion über API Gateway und gilt nicht für andere AWS Dienstaktionen, einschließlich Lambda-Aktionen außer der funktionsaufrufenden Aktion.
- `HTTP`: Dieser Integrationstyp stellt eine API als HTTP-Endpunkte im Backend dar. Bei diesem, auch als benutzerdefinierte HTTP-Integration bezeichneten `HTTP`-Integrationstyp, müssen Sie sowohl die Integrationsanforderung als auch die Integrationsantwort konfigurieren. Sie müssen die erforderlichen Daten-Mappings von der Methodenanforderung zur Integrationsanforderung und von der Integrationsantwort zur Methodenantwort einrichten.

- `HTTP_PROXY`: Die HTTP-Proxy-Integration ermöglicht einem Client den Zugriff auf HTTP-Endpunkte des Backend über eine optimierte Integrationseinrichtung mit einer einzigen API-Methode. Sie legen keine Integrationsanforderung oder Integrationsantwort fest. Amazon API Gateway übergibt die eingehende Anfrage des Clients an den HTTP-Endpunkt und übergibt die ausgehende Antwort vom HTTP-Endpunkt an den Client.
- `MOCK`: Bei diesem Integrationstyp kann API Gateway eine Antwort zurückgeben, ohne die Anfrage weiter an das Backend zu senden. Dies ist für API-Test hilfreich, da Sie so die Integrationseinrichtung testen können, ohne dass Ihnen für die Nutzung des Backends Kosten entstehen, oder gemeinsam eine API entwickeln können.

  Bei der gemeinsamen Entwicklung kann ein Team den Entwicklungsvorgang isolieren, indem es Simulationen von API-Komponenten anderer Teams über `MOCK`-Integrationen einrichtet. Dieser Integrationstyp wird auch verwendet, um CORS-Header zum Zulassen des CORS-Zugriffs durch die API-Methode zurückzugeben. Die API Gateway-Konsole integriert die `OPTIONS`-Methode zur Unterstützung von CORS sogar mit einer Mock-Integration. Gateway-Antworten sind weitere Beispiele für Pseudo-Integrationen.

Einrichten der Proxy-Integration mit einer Proxy-Ressource

Um eine Proxy-Integration in eine API Gateway-API mit einer Proxy-Ressource einzurichten, führen Sie die folgenden Aufgaben aus:

- Erstellen Sie eine Proxyressource mit der gierigen Pfadvariable `{proxy+}`.
- Legen Sie auf der Proxy-Ressource die Methode `ANY` fest.
- Integrieren Sie die Ressource und die Methode mithilfe des HTTP- oder Lambda-Integrationstyps mit einem Backend.

> **Note**
>
> Gierige Pfadvariablen, `ANY`-Methoden und Proxy-Integrationstypen sind unabhängige Funktionen, auch wenn sie häufig gemeinsam verwendet werden. Sie können eine bestimmte HTTP-Methode auf einer gierigen Ressource konfigurieren oder Nicht-Proxy-Integrationstypen auf eine Proxy-Ressource anwenden.

API Gateway setzt bestimmte Beschränkungen und Limitierungen durch, wenn Methoden mit einer Lambda-Proxy-Integration oder einer HTTP-API-Integration behandelt werden. Details hierzu finden Sie unter the section called “Wichtige Hinweise”.

> **Note**
>
> Bei der Verwendung einer Proxy-Integration mit einem Pass-Through gibt API Gateway den Standard-`Content-Type:application/json`-Header zurück, wenn der Inhaltstyp eines Payloads nicht spezifiziert ist.

Eine Proxy-Ressource ist am leistungsstärksten, wenn sie mit der HTTP-Proxy-Integration oder der Lambda-Proxy-Integration in ein Backend integriert ist.

HTTP-Proxy-Integration mit einer Proxy-Ressource

Die HTTP-Proxy-Integration, die in der REST-API des API Gateways mit `HTTP_PROXY` festgelegt ist, dient der Integration einer Methodenanfrage mit einem HTTP-Backend-Endpunkt. Bei diesem Integrationstyp übergibt API Gateway die gesamte Anfrage und Antwort einfach zwischen dem Frontend und dem Backend. Dabei gelten bestimmte Einschränkungen und Begrenzungen.

> **Note**
>
> Die HTTP-Proxy-Integration unterstützt mehrwertige Header und Abfragezeichenfolgen.

Beim Anwenden der HTTP-Proxy-Integration auf eine Proxy-Ressource können Sie die API so einrichten, dass entweder ein Teil der oder die gesamte Endpunkthierarchie des HTTP-Backends mit einer einzigen Integrationseinrichtung bereitgestellt wird. Angenommen, das Backend der Website ist in mehrere Strukturknoten-Branches eines Stammknotens (`/site`) aufgeteilt: `/site/`a_0`/`a_1`/.../`a_N, `/site/`b_0`/`b_1`/.../`b_M usw. Wenn Sie die Methode `ANY` für eine Proxy-Ressource von `/api/{proxy+}` mit den Backend-Endpunkten mit den URL-Pfaden von `/site/{proxy}` integrieren, kann eine einzelne Integrationsanforderung alle HTTP-Vorgänge (GET, POST usw.) für beliebige von $[a_0, a_1, ..., a_N, b_0, b_1, ...b_M, ...]$ unterstützen. Wenn Sie eine Proxy-Integration stattdessen auf eine bestimmte HTTP-Methode anwenden, z. B. `GET`, funktioniert die resultierende Integrationsanforderung mit den angegebenen Vorgängen (d. h. `GET`) auf jedem dieser Backend-Knoten.

Lambda-Proxy-Integration mit einer Proxy-Ressource

Die Lambda-Proxy-Integration, in der API Gateway-REST-API mit `AWS_PROXY` festgelegt ist, dient der Integration einer Methodenanfrage mit einer Lambda-Funktion im Backend. Bei diesem Integrationstyp wendet API Gateway eine Standard-Mapping-Vorlage, um die gesamte Anfrage an die Lambda-Funktion zu senden, und wandelt die Ausgabe der Lambda-Funktion in HTTP-API-Antworten um.

Ebenso können Sie die Lambda-Proxy-Integration auf eine Proxy-Ressource von `/api/{proxy+}` anwenden, um eine einzelne Integration einzurichten, sodass eine Backend-Lambda-Funktion individuell auf Änderungen an einer der API-Ressourcen unter `/api` reagiert.

API-Integrationsanfrage über die API Gateway-Konsole einrichten

Die Einrichtung einer API-Methode definiert die Methode und beschreibt ihr Verhalten. Um eine Methode einzurichten, müssen Sie Folgendes angeben: eine Ressource, einschließlich des Roots (`GET`/`POST`), auf der die Methode bereitgestellt wird, eine HTTP-Methode (GET, POST usw.) und wie diese in das Ziel-Backend integriert wird. Die Methodenanforderung und -antwort geben die Regeln für den Austausch mit der aufrufenden App vor. Sie legen fest, welche Parameter die API empfangen kann und wie die Antwort aussieht.

Die folgenden Verfahren beschreiben, wie Sie die API Gateway Gateway-Konsole verwenden, um eine Integrationsanfrage zu erstellen.

Themen

- Eine Lambda-Integration einrichten
- Richten Sie eine HTTP-Integration ein
- Richten Sie eine AWS Serviceintegration ein
- Richten Sie eine Scheinintegration ein

Eine Lambda-Integration einrichten

Verwenden Sie eine Lambda-Funktionsintegration, um Ihre API in eine Lambda-Funktion zu integrieren. Auf API-Ebene ist dies ein `AWS`-Integrationstyp, wenn Sie eine Nicht-Proxy-Integration erstellen, oder ein `AWS_PROXY`-Integrationstyp, wenn Sie eine Proxy-Integration erstellen.

So richten Sie eine Lambda-Integration ein

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Für Integration type (Integrationstyp) wählen Sie Lambda Function (Lambda-Funktion) aus.
4. Um eine Lambda-Proxyintegration zu verwenden, aktivieren Sie Lambda proxy integration (Lambda-Proxyintegration). Weitere Informationen zum Erstellen von Lambda-Proxy-Integrationen finden Sie unter the section called “ Lambda-Proxy-Integration kennenlernen ”.
5. Geben Sie für Lambda function (Lambda-Funktion) den Namen der Funktion ein.

   Wenn Sie eine Lambda-Funktion in einer anderen Region als Ihrer API verwenden, wählen Sie die Region aus dem Dropdownmenü aus und geben Sie den Namen der Lambda-Funktion ein. Wenn Sie eine kontoübergreifende Lambda-Funktion verwenden, geben Sie die Funktion ARN ein.
6. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen `50` und `29000` Millisekunden ein.
7. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 3 vonthe section called “Bearbeiten Sie eine Methodenanforderung in der Konsole”.

   Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.
8. Wählen Sie Methode erstellen aus.

Richten Sie eine HTTP-Integration ein

Verwenden Sie eine HTTP-Integration, um Ihre API in einen HTTP-Endpunkt zu integrieren. Auf der API-Ebene ist dies der `HTTP` Integrationstyp.

Um eine HTTP-Integration einzurichten

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie als Integrationstyp HTTP aus.
4. Um eine HTTP-Proxyintegration zu verwenden, aktivieren Sie HTTP proxy integration (HTTP-Proxyintegration). Weitere Informationen zu HTTP-Proxyintegrationen finden Sie unter the section called “HTTP-Proxy-Integrationen in API Gateway einrichten”.
5. Wählen Sie als HTTP method die HTTP-Methode aus, die am ehesten der Methode im HTTP-Backend entspricht.
6. Geben Sie als Endpoint URL (Endpunkt-URL) die URL des HTTP-Backends ein, das diese Methode verwenden soll.
7. Wählen Sie für Content handling (Inhaltsbehandlung) ein Verhalten bei der Inhaltsbehandlung aus.
8. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen `50` und `29000` Millisekunden ein.
9. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 3 vonthe section called “Bearbeiten Sie eine Methodenanforderung in der Konsole”.

   Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.
10. Wählen Sie Methode erstellen aus.

Richten Sie eine AWS Serviceintegration ein

Verwenden Sie eine AWS Serviceintegration, um Ihre API direkt in einen AWS Dienst zu integrieren. Auf der API-Ebene ist dies der `AWS` Integrationstyp.

Gehen Sie zum Einrichten einer API Gateway wie folgt vor:

- Erstellen Sie eine neue Lambda-Funktion.
- Legen Sie eine Ressourcenberechtigung für die Lambda-Funktion fest.
- Führen Sie alle anderen Lambda-Serviceaktionen aus.

Sie müssen den AWS -Service wählen.

Um eine AWS Serviceintegration einzurichten

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie als Integrationstyp die Option AWS Service aus.
4. Wählen Sie für AWS Region die AWS Region aus, die diese Methode zum Aufrufen der Aktion verwenden soll.
5. Wählen Sie für AWS Service den AWS Dienst aus, den Sie mit dieser Methode aufrufen möchten.
6. Geben Sie für AWS Subdomain die Subdomain ein, die AWS vom Service verwendet wird. Normalerweise bleibt dieses Feld leer. Einige AWS -Services können Subdomänen als Teil des Hosts unterstützen. Weitere Informationen über die Verfügbarkeit und gegebenenfalls Details finden Sie in der Service-Dokumentation.
7. Wählen Sie als HTTP method die der Aktion entsprechende HTTP-Methode aus. Informationen zum HTTP-Methodentyp finden Sie in der API-Referenzdokumentation für den AWS Dienst, den Sie für den Dienst ausgewählt haben.AWS
8. Wählen Sie unter Action type (Aktionstyp) entweder Use action name (Aktionsnamen verwenden) aus, um eine API-Aktion zu verwenden, oder Use path override (Pfadüberschreibung verwenden), um einen benutzerdefinierten Ressourcenpfad zu verwenden. Verfügbare Aktionen und benutzerdefinierte Ressourcenpfade finden Sie in der API-Referenzdokumentation für den AWS Dienst, den Sie für den AWS Dienst ausgewählt haben.
9. Geben Sie entweder einen Action name (Aktionsnamen) oder eine Path override (Pfadüberschreibung) ein.
10. Geben Sie unter Execution Role (Ausführungsrolle) den ARN der IAM-Rolle ein, die die Methode zum Aufrufen der Aktion verwendet.

    Zum Erstellen einer IAM-Rolle können Sie die Anweisungen in the section called “Schritt 1: Erstellen Sie die AWS Service-Proxy-Ausführungsrolle” anpassen. Legen Sie eine

Zugriffsrichtlinie mit der gewünschten Anzahl an Aktions- und Ressourcenanweisungen im folgenden Format fest:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "action-statement"
      ],
      "Resource": [
        "resource-statement"
      ]
    },
    ...
  ]
}
```

Die Syntax der Aktions- und Ressourcenanweisungen finden Sie in der Dokumentation des AWS Dienstes, den Sie für den AWS Dienst ausgewählt haben.

Für die Vertrauensbeziehung der IAM-Rolle geben Sie Folgendes an, wodurch API Gateway in der Lage ist, im Namen Ihres AWS -Kontos aktiv zu werden:

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

11. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout

(Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen `50` und `29000` Millisekunden ein.

12. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 3 vonthe section called “Bearbeiten Sie eine Methodenanforderung in der Konsole”.

    Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

13. Wählen Sie Methode erstellen aus.

Richten Sie eine Scheinintegration ein

Verwenden Sie eine Scheinintegration, wenn Sie möchten, dass API Gateway als Ihr Backend fungiert, um statische Antworten zurückzugeben. Auf der API-Ebene ist dies der `MOCK` Integrationstyp. Normalerweise können Sie die `MOCK` Integration verwenden, wenn Ihre API noch nicht finalisiert ist, Sie aber API-Antworten erzeugen möchten, um abhängigen Teams Tests zu ermöglichen. Für die `OPTION`-Methode legt API Gateway die `MOCK`-Integration als Standard fest, um CORS-fähige Header für die angewandte API-Ressource zurückzugeben.

Um eine Scheinintegration einzurichten

1. Wählen Sie im Bereich Resources (Ressourcen) die Option Create method (Methode erstellen) aus.
2. Wählen Sie als Method type (Methodentyp) eine HTTP-Methode aus.
3. Wählen Sie als Integrationstyp die Option Mock aus.
4. (Optional) Sie können die Einstellungen für die Methodenanforderung mithilfe der folgenden Dropdownmenüs konfigurieren. Wählen Sie Einstellungen für Methodenanfragen und konfigurieren Sie Ihre Methodenanforderung. Weitere Informationen finden Sie in Schritt 3 vonthe section called “Bearbeiten Sie eine Methodenanforderung in der Konsole”.

    Sie können Ihre Einstellungen für Methodenanfragen auch konfigurieren, nachdem Sie Ihre Methode erstellt haben.

5. Wählen Sie Methode erstellen aus.

Integrationsantwort in API Gateway einrichten

Bei einer Nicht-Proxy-Integration müssen Sie mindestens eine Integrationsantwort einrichten und als Standardantwort festlegen, um das Ergebnis vom Backend an den Client zu übergeben. Sie können das Ergebnis unverändert übergeben oder die Integrationsantwortdaten bei unterschiedlichem Format in die Methodenantwortdaten umwandeln.

Bei einer Proxy-Integration leitet API Gateway die Backend-Ausgabe automatisch als HTTP-Antwort an den Client weiter. Sie legen weder Integrationsantwort noch Methodenantwort fest.

Führen Sie die folgenden erforderlichen und optionalen Schritte aus, um eine Integrationsantwort einzurichten:

1. Geben Sie einen HTTP-Statuscode einer Methodenantwort an, der die Integrationsantwortdaten zugeordnet sind. Diese Information ist erforderlich.
2. Definieren Sie einen regulären Ausdruck, um Backend-Ausgaben als Integrationsantwort auszuwählen. Wenn Sie hier keine Angaben machen, wird als Antwort die Standardantwort verwendet, die immer dann verwendet wird, wenn keine anderweitige Antwort konfiguriert ist.
3. Deklarieren Sie gegebenenfalls Mappings bestehend aus Schlüssel-Wert-Paaren, um angegebene Integrationsantwortparameter den gegebenen Methodenantwortparametern zuzuordnen.
4. Fügen Sie gegebenenfalls Inhalts-Mapping-Vorlagen hinzu, um vorhandene Integrationsantwortnutzlasten in angegebene Methodenantwortnutzlasten umzuwandeln.
5. Legen Sie gegebenenfalls fest, wie Typumwandlungen für binäre Nutzlasten verarbeitet werden sollen.

Die Integrationsantwort ist eine HTTP-Antwort, die die Backend-Antwort kapselt. Bei HTTP-Endpunkten ist die Backend-Antwort eine HTTP-Antwort. Der Statuscode der Integrationsantwort kann den vom Backend zurückgegebenen Statuscode übernehmen. Der Inhalt der Integrationsantwort ist die vom Backend zurückgegebene Nutzlast. Bei einem Lambda-Endpunkt ist die Backend-Antwort die von der Lambda-Funktion zurückgegebene Ausgabe. Bei der Lambda-Integration wird die Ausgabe der Lambda-Funktion als `200 OK`-Antwort zurückgegeben. Die Nutzlast kann das Ergebnis als JSON-Daten enthalten, einschließlich einer JSON-Zeichenfolge oder einem JSON-Objekt oder einer Fehlermeldung als JSON-Objekt. Sie können der Eigenschaft selectionPattern einen regulären Ausdruck zuweisen, um eine Fehlerantwort einer passenden HTTP-Fehlerantwort zuzuordnen. Weitere Informationen über die Fehlerantwort der Lambda-Funktion finden Sie unter Lambda-Fehler in API Gateway behandeln. Bei der Lambda-Proxy-Integration muss die Lambda-Funktion eine Ausgabe im folgenden Format zurückgeben:

```
{
    statusCode: "...",            // a valid HTTP status code
    headers: {
        custom-header: "..."      // any API-specific custom header
    },
    body: "...",                  // a JSON string.
    isBase64Encoded:  true|false  // for binary support
}
```

Es besteht keine Notwendigkeit, die Antwort der Lambda-Funktion der richtigen HTTP-Antwort zuzuordnen.

Um das Ergebnis an den Client zurückzugeben, richten Sie die Integrationsantwort so ein, dass die Endpunktantwort unverändert an die entsprechende Methodenantwort übergeben wird. Alternativ können Sie die Endpunktantwortdaten auch den Methodenantwortdaten zuordnen. Es können folgende Antwortdaten zugeordnet werden: der Antwortstatuscode, die Parameter des Antwort-Headers und der Antwortinhalt. Wenn keine Methodenantwort für den zurückgegebenen Statuscode definiert ist, gibt API Gateway einen 500-Fehler zurück. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.

Lambda-Integrationen in API Gateway einrichten

Sie können eine API-Methode mit einer Lambda-Funktion integrieren, indem Sie eine Lambda-Proxy-Integration oder eine nicht-proxy (benutzerdefinierte) Lambda-Integration verwenden.

Bei der Lambda-Proxy-Integration ist die erforderliche Einrichtung einfach. Legen Sie die HTTP-Methode der Integration auf POST und den Integrationsendpunkt-URI auf den ARN der Lambda-Funktion-Aufrufsaktion einer bestimmten Lambda-Funktion fest. Gewähren Sie dann API Gateway die Berechtigung zum Aufrufen der Lambda-Funktion in Ihrem Namen.

In der Lambda-Nicht-Proxy-Integration müssen Sie zusätzlich zu den Einrichtungsschritten der Proxy-Integration angeben, wie die eingehenden Anfragedaten der Integrationsanforderung und die resultierenden Integrationsantwortdaten der Methodenantwort zugeordnet werden.

Themen

- Lambda-Proxy-Integrationen in API Gateway einrichten
- Benutzerdefinierte Lambda-Integrationen in API Gateway einrichten

- Asynchronen Aufruf der Backend-Lambda-Funktion einrichten
- Lambda-Fehler in API Gateway behandeln

Lambda-Proxy-Integrationen in API Gateway einrichten

Themen

- API Gateway Lambda-Proxy-Integration kennenlernen
- Support für mehrwertige Header- und Abfragezeichenfolgenparameter
- Proxy-Ressource mit Lambda-Proxy-Integration einrichten
- Richten Sie die Lambda-Proxyintegration mit dem ein AWS CLI
- Eingabeformat einer Lambda-Funktion für die Proxy-Integration
- Ausgabeformat einer Lambda-Funktion für die Proxy-Integration

API Gateway Lambda-Proxy-Integration kennenlernen

Die Amazon API Gateway-Lambda-Proxy-Integration ist ein einfacher, leistungsstarker und schneller Mechanismus zur Erstellung einer API mit der Einrichtung einer einzigen API-Methode. Die Lambda-Proxy-Integration ermöglicht es dem Client, eine einzelne Lambda-Funktion im Backend aufzurufen. Die Funktion greift auf viele Ressourcen oder Funktionen anderer AWS Dienste zu, einschließlich des Aufrufs anderer Lambda-Funktionen.

Bei der Lambda-Proxy-Integration übergibt API Gateway beim Senden einer API-Anfrage ein Ereignisobjekt an die integrierte Lambda-Funktion. Allerdings wird die Reihenfolge der Anfrageparameter nicht beibehalten. Diese Anforderungsdaten umfassen die Abfrage-Header, Parameter für Abfragezeichenfolgen, die URL-Pfadvariablen, die Nutzlast und API-Konfigurationsdaten. Die Konfigurationsdaten können den aktuellen Bereitstellungsstufennamen, Stufenvariablen, die Benutzeridentität oder den Autorisierungskontext (falls vorhanden) umfassen. Die Backend-Lambda-Funktion analysiert die eingehenden Anfragedaten, um die eigene Antwort zu bestimmen. Damit API Gateway die Lambda-Ausgabe als API-Antwort an den Client weiterleiten kann, muss die Lambda-Funktion das Ergebnis in diesem Format zurückgeben.

Da API Gateway für die Lambda-Proxy-Integration zwischen dem Client und der Backend-Lambda-Funktion nur unwesentlich eingreift, können sich der Client und die integrierte Lambda-Funktion an Änderungen in der jeweils anderen Funktion anpassen, ohne die bestehende Integrationseinrichtung der API zu stören. Hierzu muss der Client den Anwendungsprotokollen der Backend-Lambda-Funktion folgen.

Sie können eine Lambda-Proxy-Integration für jede API-Methode einrichten. Aber eine Lambda-Proxy-Integration ist leistungsstärker, wenn sie über eine allgemeine Proxy-Ressource für eine API-Methode konfiguriert wird. Die allgemeine Proxy-Ressource kann durch eine spezielle Vorlagenpfadvariable `{proxy+}`, den Catch-All-`ANY`-Methodenplatzhalter oder beides angegeben werden. Der Client kann die Eingabe an die Backend-Lambda-Funktion in der eingehenden Anfrage als Anfrageparameter oder als entsprechende Nutzlast übergeben. Zu den Anforderungsparametern gehören Header-, URL-Pfadvariablen, Abfragezeichenfolgenparameter und die entsprechende Nutzlast. Die integrierte Lambda-Funktion prüft alle Eingangsquellen vor ihrer Verarbeitung und der Antwort an den Client mit sinnvollen Fehlermeldungen, falls die erforderliche Eingabe fehlt.

Beim Aufrufen einer mit der allgemeinen HTTP-Methode `ANY` und der allgemeinen Ressource `{proxy+}` integrierten API-Methode sendet der Client eine Anforderung mit einer bestimmten HTTP-Methode statt mit `ANY`. Der Client gibt zudem einen bestimmten URL-Pfad statt `{proxy+}` an und umfasst alle erforderlichen Header, Abfragezeichenfolgenparameter oder eine anwendbare Nutzlast.

Die folgende Liste enthält eine Zusammenfassung des Laufzeitverhaltens verschiedener API-Methoden bei der Lambda-Proxy-Integration:

- `ANY /{proxy+}`: Der Client muss eine bestimmte HTTP-Methode auswählen, eine bestimmte Ressourcenpfadhierarchie festlegen und kann alle Header, Parameter für Abfragezeichenfolgen und den Payload für die Eingabe an die integrierte Lambda-Funktion festlegen.
- `ANY /res`: Der Client muss eine bestimmte HTTP-Methode auswählen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabe an die integrierte Lambda-Funktion festlegen.
- `GET|POST|PUT|... /{proxy+}`: Der Client kann eine bestimmte Ressourcenpfadhierarchie festlegen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabe an die integrierte Lambda-Funktion festlegen.
- `GET|POST|PUT|... /res/{path}/...`: Der Client muss ein bestimmtes Pfadsegment (für die `{path}`-Variable) festlegen und kann alle Anfrage-Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabedaten an die integrierte Lambda-Funktion festlegen.
- `GET|POST|PUT|... /res`: Der Client kann bestimmte Anfrage-Header auswählen und kann alle Header, Parameter für Abfragezeichenfolgen und die Nutzlast für die Eingabedaten an die integrierte Lambda-Funktion festlegen.

Sowohl die Proxy-Ressource `{proxy+}` und die benutzerdefinierte Ressource `{custom}` werden als Vorlagenpfadvariablen ausgedrückt. Allerdings kann `{proxy+}` auf alle Ressourcen in

einer Pfadhierarchie verweisen, während sich `{custom}` nur auf ein bestimmtes Pfadsegment bezieht. Beispielsweise könnte ein Lebensmittelgeschäft sein Online-Produktsortiment nach Abteilungsnamen, Kategorien und Produktarten organisieren. Die Website des Lebensmittelgeschäfts kann dann verfügbare Produkte über die folgenden Vorlagenpfadvariablen darstellen: `/{department}/{produce-category}/{product-type}`. Zum Beispiel werden Äpfel mit `/produce/fruit/apple` und Karotten mit `/produce/vegetables/carrot` dargestellt. Es kann auch `/{proxy+}` als Platzhalter für eine beliebige Abteilung, eine beliebige Kategorie oder einen Produkttyp verwendet werden, nach dem ein Kunde im Online-Shop suchen kann. Beispielsweise kann `/{proxy+}` auf eines der folgenden Elemente verweisen:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Um Kunden nach den verfügbaren Produkten, der Kategorie und der zugeordneten Abteilung suchen zu lassen, können Sie eine einzelne `GET /{proxy+}`-Methode mit Leseberechtigungen bereitstellen. Um einem Supervisor das Aktualisieren des `produce` Abteilungsbestands zu ermöglichen, können Sie eine andere, einzelne `PUT /produce/{proxy+}`-Methode mit Lese-/Schreibberechtigungen hinzufügen. Um einem Kassierer das Aktualisieren der Gesamtsumme einer Gemüsesorte zu ermöglichen, können Sie eine `POST /produce/vegetables/{proxy+}`-Methode mit Lese-/Schreibberechtigungen einrichten. Wenn ein Shop-Manager sämtliche möglichen Aktionen für beliebige verfügbare Produkte durchführen soll, kann der Online-Shop-Entwickler die `ANY /{proxy+}` Methode mit Lese-/Schreibberechtigungen hinzufügen. In jedem Fall muss der Kunde oder der Mitarbeiter zur Laufzeit ein bestimmtes Produkt eines bestimmten Typs in einer bestimmten Abteilung, eine bestimmte Kategorie in einer bestimmten Abteilung, oder eine bestimmte Abteilung auswählen.

Weitere Informationen zum Einrichten der API Gateway-Proxy-Integrationen finden Sie unter Einrichten der Proxy-Integration mit einer Proxy-Ressource.

Die Proxy-Integration erfordert, dass der Client über detaillierte Kenntnisse der Backend-Anforderungen verfügt. Um eine optimale App-Leistung und Benutzererfahrung sicherzustellen, muss der Backend-Entwickler die Anforderungen des Backends klar an den Client-Entwickler kommunizieren. Er muss einen robusten Feedback-Mechanismus für nicht erfüllte Anforderungen bereitstellen.

Support für mehrwertige Header- und Abfragezeichenfolgenparameter

API Gateway unterstützt mehrere gleichnamige Header und Abfragezeichenfolgenparameter. Mehrwertige Header sowie einwertige Header und Parameter können in denselben Anforderungen und Antworten kombiniert werden. Weitere Informationen finden Sie unter Eingabeformat einer Lambda-Funktion für die Proxy-Integration und Ausgabeformat einer Lambda-Funktion für die Proxy-Integration.

Proxy-Ressource mit Lambda-Proxy-Integration einrichten

Erstellen Sie zum Einrichten einer Proxy-Ressource mit dem Lambda-Proxy-Integrationstyp eine API-Ressource mit einem Greedy-Pfadparameter (z. B. `/parent/{proxy+}`) und integrieren Sie diese Ressource mit einem Lambda-Funktions-Backend (z. B. `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource`) für die Methode `ANY`. Der Greedy-Pfadparameter muss am Ende des API-Ressourcenpfads stehen. Wie bei einer Nicht-Proxy-Ressource können Sie die Proxy-Ressource einrichten, indem Sie die API Gateway-Konsole verwenden, eine OpenAPI-Definitionsdatei importieren oder die API Gateway-REST-API direkt aufrufen.

Die folgende OpenAPI-API-Definitionsdatei zeigt ein Beispiel für eine API mit einer Proxyressource, die mit der Lambda-Funktion `SimpleLambda4ProxyResource` integriert wird.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T17:50:37Z",
      "title": "ProxyIntegrationWithLambda"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
                  }
```

```
               }
            ],
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
                  "default": {
                     "statusCode": "200"
                  }
               },
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "POST",
               "cacheNamespace": "roq9wj",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "aws_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://gy415nuibc.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/testStage"
            }
         }
      }
   ]
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T17:50:37Z",
    "title": "ProxyIntegrationWithLambda"
  },
```

```
  "host": "gy415nuibc.execute-api.us-east-1.amazonaws.com",
  "basePath": "/testStage",
  "schemes": [
    "https"
  ],
  "paths": {
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "cacheNamespace": "roq9wj",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "aws_proxy"
        }
      }
    }
  }
}
```

Bei der Lambda-Proxy-Integration ordnet API Gateway zur Laufzeit eine eingehende Anfrage dem Eingabe-`event`-Parameter der Lambda-Funktion zu. Die Eingabe umfasst die Anforderungsmethode, den Pfad, Header, jegliche Zeichenparameter, jegliche Nutzlast, zugehörigen Kontext und alle definierten Stufenvariablen. Eine Beschreibung des Eingabeformats finden Sie unter Eingabeformat einer Lambda-Funktion für die Proxy-Integration. Damit API Gateway die Lambda-Ausgabe erfolgreich den HTTP-API-Antworten zuordnen kann, muss die Lambda-Funktion das Ergebnis in dem unter Ausgabeformat einer Lambda-Funktion für die Proxy-Integration beschriebenen Format ausgeben.

Bei der Lambda-Proxy-Integration einer Proxy-Ressource durch die `ANY`-Methode dient die einzelne Backend-Lambda-Funktion als Ereignis-Handler für alle Anfragen über die Proxy-Ressource. Beispielsweise können Sie zum Protokollieren der Datenverkehrsmuster ein Mobilgerät veranlassen, seinen Standort mit Angaben zu Bundesland, Stadt, Straße und Gebäude zu senden, indem Sie eine Anforderung mit `/state/city/street/house` im URL-Pfad für die Proxy-Ressource übermitteln. Die Backend-Lambda-Funktion kann dann den URL-Pfad analysieren und die Location-tuples in eine DynamoDB-Tabelle einfügen.

Richten Sie die Lambda-Proxyintegration mit dem ein AWS CLI

In diesem Abschnitt zeigen wir, wie Sie AWS CLI eine API mit der Lambda-Proxyintegration einrichten.

> **Note**
>
> Detaillierte Anweisungen zur Verwendung der API Gateway-Konsole zur Konfiguration einer Proxy-Ressource mit der Lambda-Proxy-Integration finden Sie unter Tutorial: Hello World REST-API mit Lambda-Proxy-Integration erstellen.

Als Beispiel verwenden wir die folgende Lambda-Beispielfunktion als Backend der API:

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

```
    if (event.greeter && event.greeter!=="") {
        greeter =  event.greeter;
    } else if (event.body && event.body !== "") {
        var body = JSON.parse(event.body);
        if (body.greeter && body.greeter !== "") {
            greeter = body.greeter;
        }
    } else if (event.queryStringParameters && event.queryStringParameters.greeter &&
 event.queryStringParameters.greeter !== "") {
        greeter = event.queryStringParameters.greeter;
    } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
 event.multiValueHeaders.greeter != "") {
        greeter = event.multiValueHeaders.greeter.join(" and ");
    } else if (event.headers && event.headers.greeter && event.headers.greeter != "") {
        greeter = event.headers.greeter;
    }

    res.body = "Hello, " + greeter + "!";
    callback(null, res);
};
```

Vergleicht man dies mit der benutzerdefinierten Lambda-Integrationseinrichtung , kann die Eingabe für diese Lambda-Funktion in den Anfrageparametern und im Datenteil definiert werden. Sie haben mehr Spielraum, um dem Client die Weitergabe der gleichen Eingabedaten zu ermöglichen. Hier kann der Client den Namen von "greeter" als Abfragezeichenfolgeparameter, Kopfzeile oder Texteigenschaft übergeben. Die Funktion kann außerdem die benutzerdefinierte Lambda-Integration unterstützen. Die API-Einrichtung ist einfacher. Sie müssen keine Methoden- oder Integrationsantwort konfigurieren.

So richten Sie eine Lambda-Proxyintegration mit dem AWS CLI

1. Rufen Sie den Befehl `create-rest-api` auf, um eine API zu erstellen:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
   ```

   Notieren Sie den in der Antwort für die API-`id` angegebenen Wert (`te6si5ach7`).

   ```
   {
       "name": "HelloWorldProxy (AWS CLI)",
       "id": "te6si5ach7",
       "createdDate": 1508461860
   ```

```
}
```

Sie benötigen die API-`id` in diesem Abschnitt.

2. Rufen Sie den Befehl `get-resources` auf, um die `id` der Stammressource zu ermitteln.

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

Die erfolgreiche Antwort wird wie folgt angezeigt:

```
{
    "items": [
        {
            "path": "/",
            "id": "krznpq9xpg"
        }
    ]
}
```

Notieren Sie den `id`-Wert (`krznpq9xpg`) der Ressource. Sie benötigen diese in den nächsten Schritten.

3. Rufen Sie `create-resource` auf, um eine API Gateway-Ressource `/greeting` zu erstellen:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part {proxy+}
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Notieren Sie den ausgegebenen `id`-Wert (`2jf6xt`) der `{proxy+}`-Ressource. Sie benötigen die ID, um im nächsten Schritt eine Methode auf der `/{proxy+}`-Ressource zu erstellen.

4. Rufen Sie `put-method` zum Erstellen einer ANY-Methodenanforderung `ANY /{proxy+}` auf:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --region us-west-2 \
       --resource-id 2jf6xt \
       --http-method ANY \
       --authorization-type "NONE"
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
{
    "apiKeyRequired": false,
    "httpMethod": "ANY",
    "authorizationType": "NONE"
}
```

Diese API-Methode gestattet dem Client, einen Gruß über die Lambda-Funktion am Backend zu empfangen oder zu senden.

5. Rufen Sie `put-integration` auf, um die Integration der `ANY /{proxy+}`-Methode mit einer Lambda-Funktion namens `HelloWorld` einzurichten. Diese Funktion beantwortet die Anforderung mit der Nachricht `"Hello, {name}!"`, sofern der `greeter`-Parameter angegeben wurde, oder `"Hello, World!"`, wenn kein Abfragezeichenfolgeparameter festgelegt wurde.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method ANY \
        --type AWS_PROXY \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

> **Important**
>
> Für Lambda-Integrationen müssen Sie entsprechend der Spezifikation der Lambda-Service-Aktion für Funktionsaufrufe die HTTP-Methode `POST` für die Integrationsanfrage

verwenden. Die IAM-Rolle `apigAwsProxyRole` muss über Richtlinien verfügen, die es dem `apigateway`-Service erlauben, Lambda-Funktionen aufzurufen. Weitere Informationen zu IAM-Berechtigungen finden Sie unter the section called “ API Gateway-Berechtigungsmodell für den Aufruf einer API”.

Im Erfolgsfall sieht die Ausgabe folgendermaßen oder ähnlich aus:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:1234567890:function:HelloWorld/invocations",
    "httpMethod": "POST",
    "cacheNamespace": "vvom7n",
    "credentials": "arn:aws:iam::1234567890:role/apigAwsProxyRole",
    "type": "AWS_PROXY"
}
```

Anstatt eine IAM-Rolle für `credentials` bereitzustellen, können Sie den Befehl add-permission aufrufen, um ressourcenbasierte Berechtigungen hinzuzufügen. Dies erledigt die API Gateway-Konsole.

6. Rufen Sie `create-deployment` auf, um die API für eine `test`-Stufe bereitzustellen:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

7. Testen Sie die API mithilfe der folgenden cURL-Befehle in einem Terminal.

   API-Aufruf mit dem Abfragezeichenfolgeparameter `?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=jane'
```

   API-Aufruf mit einem Header-Parameter `greeter:jane`:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -H 'greeter: jane'
```

API-Aufruf mit einem Textkörper-Parameter `{"greeter":"jane"}`:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -d '{ "greeter": "jane" }'
```

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

```
Hello, jane!
```

Eingabeformat einer Lambda-Funktion für die Proxy-Integration

Bei der Lambda-Proxy-Integration ordnet API Gateway die gesamte Client-Anforderung dem `event`-Eingabeparameter der Backend-Lambda-Funktion zu. Das folgende Beispiel zeigt die Struktur eines Ereignisses, das API Gateway an eine Lambda-Proxy-Integration sendet.

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
```

```
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "IP",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
```

```
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

> **Note**
>
> Wählen Sie im Bereich
>
> - Der `headers`Schlüssel kann nur einwertige Header enthalten
> - Der `multiValueHeaders`Schlüssel kann mehrwertige Header sowie einwertige Header enthalten.
> - Wenn Sie Werte sowohl für `headers` als auch für `multiValueHeaders` angeben, führt API Gateway diese in einer einzigen Liste zusammen. Wenn in beiden das gleiche Schlüssel / Wert-Paar angegeben ist, werden nur die Werte von in der `multiValueHeaders`zusammengeführten Liste angezeigt.

In der Eingabe an die Backend-Lambda-Funktion ist das `requestContext`-Objekt eine Zuordnung von Schlüssel-Werte-Paaren. In jedem Paar ist der Schlüssel der Name einer $context-Variableneigenschaft, und der Wert ist der Wert dieser Eigenschaft. API Gateway fügt der Map möglicherweise neue Schlüssel hinzu.

Abhängig von den aktivierten Funktionen kann die Zuweisung `requestContext` von API zu API unterschiedlich sein. Im obigen Beispiel ist z. B. kein Autorisierungstyp angegeben, sodass keine `$context.authorizer.*`- oder `$context.identity.*`-Eigenschaften vorhanden sind. Wenn ein Autorisierungstyp angegeben wird, bewirkt dies, dass API Gateway autorisierte Benutzerinformationen wie folgt in einem `requestContext.identity`-Objekt an den Integrationsendpunkt weiterleitet:

- Wenn der Autorisierungstyp `AWS_IAM` ist, enthalten die autorisierten Benutzerinformationen `$context.identity.*`-Eigenschaften.

- Wenn der Autorisierungstyp `COGNITO_USER_POOLS` ist (Amazon Cognito-Genehmiger), enthalten die autorisierten Benutzerinformationen die Eigenschaften `$context.identity.cognito*` und `$context.authorizer.claims.*`.
- Wenn der Autorisierungstyp `CUSTOM` (Lambda-Genehmiger) ist, enthalten die autorisierten Benutzerinformationen `$context.authorizer.principalId`- und andere relevante `$context.authorizer.*`-Eigenschaften.

Ausgabeformat einer Lambda-Funktion für die Proxy-Integration

Bei der Lambda-Proxy-Integration erfordert API Gateway, dass die Backend-Lambda-Funktion die Ausgabe gemäß dem folgenden JSON-Format zurückgibt:

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
    "body": "..."
}
```

In der Konsolenausgabe:

- Die Schlüssel `headers` und `multiValueHeaders` können nicht angegeben werden, wenn keine zusätzlichen Antwortheader zurückgegeben werden sollen.
- Der `headers`Schlüssel kann nur einwertige Header enthalten
- Der `multiValueHeaders`Schlüssel kann mehrwertige Header sowie einwertige Header enthalten. Sie können den `multiValueHeaders`Schlüssel verwenden, um alle zusätzlichen Header anzugeben, einschließlich einzelner Werte.
- Wenn Sie Werte sowohl für `headers` als auch für `multiValueHeaders` angeben, führt API Gateway diese in einer einzigen Liste zusammen. Wenn in beiden das gleiche Schlüssel / Wert-Paar angegeben ist, werden nur die Werte von in der `multiValueHeaders`zusammengeführten Liste angezeigt.

Um CORS für die Lambda-Proxy-Integration zu aktivieren, müssen Sie der Ausgabe `headers` `Access-Control-Allow-Origin:`*`domain-name`* hinzufügen. *`domain-name`* kann * für einen beliebigen Domänennamen sein. Die Ausgabe `body` wird als Nutzlast der Methodenanforderung an das Frontend angeordnet Wenn `body` ein binärer Blob ist, können Sie ihn als Base64-kodierte

Zeichenfolge codieren, indem Sie `isBase64Encoded` auf `true` festlegen und `*/*` als Binary Media Type (Binärer Medientyp) konfigurieren. Andernfalls können Sie den Wert auf `false` festlegen oder keinen Wert angeben.

> **Note**
>
> Weitere Informationen zur Aktivierung der Unterstützung von Binärdateien finden Sie unter Binärunterstützung über die API Gateway-Konsole aktivieren. Eine Beispielfunktion für Lambda finden Sie unter Rückgabe binärer Medien von einer Lambda-Proxy-Integration.

Wenn die Funktionsausgabe ein anderes Format hat, gibt API Gateway eine `502 Bad Gateway`-Fehlerantwort zurück.

Um eine Antwort in einer Lambda-Funktion in Node.js zurückzugeben, können Sie Befehle wie den folgenden verwenden:

- Um ein erfolgreiches Ergebnis zurückzugeben, rufen Sie au `callback(null, {"statusCode": 200, "body": "results"})`.
- Rufen Sie zum Auslösen einer Ausnahme au `callback(new Error('internal server error'))`.
- Bei einem Fehler auf Clientseite, z. B. wenn ein erforderlicher Parameter fehlt, können Sie `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` aufrufen, um den Fehler zurückzugeben, ohne eine Ausnahme auszulösen.

In einer Lambda-`async`-Funktion in Node.js würde die entsprechende Syntax so lauten:

- Um ein erfolgreiches Ergebnis zurückzugeben, rufen Sie au `return {"statusCode": 200, "body": "results"}`.
- Rufen Sie zum Auslösen einer Ausnahme au `throw new Error("internal server error")`.
- Bei einem Fehler auf Clientseite, z. B. wenn ein erforderlicher Parameter fehlt, können Sie `return {"statusCode": 400, "body": "Missing parameters of ..."}` aufrufen, um den Fehler zurückzugeben, ohne eine Ausnahme auszulösen.

Benutzerdefinierte Lambda-Integrationen in API Gateway einrichten

Um zu zeigen, wie die benutzerdefinierte Lambda-Integration eingerichtet wird, erstellen wir ein API Gateway-API, um die `GET /greeting?greeter={name}`-Methode zum Aufrufen einer Lambda-Funktion bereitzustellen. Verwenden Sie eine der folgenden Lambda-Beispielfunktionen für Ihre API.

Verwenden Sie eine der folgenden Lambda-Beispielfunktionen:

Node.js

```
export const handler = function(event, context, callback) {
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    };
    if (event.greeter==null) {
        callback(new Error('Missing the required greeter parameter.'));
    } else if (event.greeter === "") {
        res.body = "Hello, World";
        callback(null, res);
    } else {
        res.body = "Hello, " + event.greeter +"!";
        callback(null, res);
    }
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)
    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
```

```
        res['body'] = "Hello, World"
    elif (event['greeter']):
        res['body'] = "Hello, " + event['greeter'] + "!"
    else:
        raise Exception('Missing the required greeter parameter.')

    return res
```

Die Funktion gibt die Nachricht "`Hello, {name}!`" zurück, sofern der `greeter`-Parameterwert keine leere Zeichenfolge ist. Wenn der "`Hello, World!`"-Wert eine leere Zeichenfolge ist, wird als Antwort `greeter` zurückgegeben. Die Funktion gibt eine Fehlermeldung zurück, "`Missing the required greeter parameter.`" wenn der "greeter"-Parameter in der eingehenden Anforderung nicht festgelegt ist. Wir geben der Funktion den Namen `HelloWorld`.

Sie können sie in der Lambda-Konsole oder mithilfe der AWS CLI erstellen. In diesem Abschnitt definieren wir diese Funktion mithilfe des folgenden ARN:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Wenn die Lambda-Funktion im Backend eingerichtet ist, fahren Sie mit dem Einrichten der API fort.

Um die benutzerdefinierte Lambda-Integration einzurichten, verwenden Sie AWS CLI

1. Rufen Sie den Befehl `create-rest-api` auf, um eine API zu erstellen:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
   ```

   Notieren Sie den in der Antwort für die API-`id` angegebenen Wert (`te6si5ach7`).

   ```
   {
       "name": "HelloWorld (AWS CLI)",
       "id": "te6si5ach7",
       "createdDate": 1508461860
   }
   ```

   Sie benötigen die API-`id` in diesem Abschnitt.
2. Rufen Sie den Befehl `get-resources` auf, um die `id` der Stammressource zu ermitteln.

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

Die erfolgreiche Antwort lautet folgendermaßen:

```
{
    "items": [
        {
            "path": "/",
            "id": "krznpq9xpg"
        }
    ]
}
```

Notieren Sie den `id`-Wert (`krznpq9xpg`) der Ressource. Sie benötigen diese in den nächsten Schritten.

3. Rufen Sie `create-resource` auf, um eine API Gateway-Ressource `/greeting` zu erstellen:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part greeting
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
{
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Notieren Sie den ausgegebenen `id`-Wert (`2jf6xt`) der `greeting`-Ressource. Sie benötigen die ID, um im nächsten Schritt eine Methode auf der `/greeting`-Ressource zu erstellen.

4. Rufen Sie `put-method` zum Erstellen einer `GET /greeting?greeter={name}`-API-Methodenanforderung auf:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --region us-west-2 \
```

```
        --resource-id 2jf6xt \
        --http-method GET \
        --authorization-type "NONE" \
        --request-parameters method.request.querystring.greeter=false
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
        "method.request.querystring.greeter": false
    }
}
```

Diese API-Methode gestattet dem Client, einen Gruß über die Lambda-Funktion am Backend zu empfangen. Der `greeter`-Parameter ist optional, da das Backend entweder einen anonymen Aufrufer oder einen selbst identifizierten Aufrufer verarbeiten sollte.

5. Rufen Sie `put-method-response` auf, um auf die Anforderung von `200 OK` die Antwort `GET /greeting?greeter={name}` einzurichten:

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --status-code 200
```

6. Rufen Sie `put-integration` auf, um die Integration der `GET /greeting?greeter={name}`-Methode mit einer Lambda-Funktion namens `HelloWorld` einzurichten. Die Funktion beantwortet die Anforderung mit der Nachricht `"Hello, {name}!"`, sofern der `greeter`-Parameter angegeben wurde, oder `"Hello, World!"`, wenn kein Abfragezeichenfolgeparameter festgelegt wurde.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
```

```
        --http-method GET \
        --type AWS \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \
        --request-templates '{"application/json":"{\"greeter\":
\"$input.params('greeter')\"}"}' \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Die hier bereitgestellte Mapping-Vorlage übersetzt den `greeter`-Abfragezeichenfolgeparameter in den `greeter`-Parameter der JSON-Nutzlast. Dies ist notwendig, da die Eingabe einer Lambda-Funktion im Textkörper ausgedrückt werden muss.

> **Important**
>
> Für Lambda-Integrationen müssen Sie entsprechend der Spezifikation der Lambda-Service-Aktion für Funktionsaufrufe die HTTP-Methode `POST` für die Integrationsanfrage verwenden. Der `uri`-Parameter ist der ARN der Aktion, die die Funktion aufruft.
> Im Erfolgsfall sollte die Ausgabe in etwa folgendermaßen aussehen:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
    "httpMethod": "POST",
    "requestTemplates": {
        "application/json": "{\"greeter\":\"$input.params('greeter')\"}"
    },
    "cacheNamespace": "krznpq9xpg",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "type": "AWS"
}
```

Die IAM-Rolle `apigAwsProxyRole` muss über Richtlinien verfügen, die es dem `apigateway`-Service erlauben, Lambda-Funktionen aufzurufen. Anstatt eine IAM-Rolle für `credentials` bereitzustellen, können Sie den Befehl add-permission aufrufen, um ressourcenbasierte Berechtigungen hinzuzufügen. Auf diese Weise fügt die API Gateway-Konsole diese Berechtigungen hinzu.

7. Rufen Sie `put-integration-response` auf, um die Integrationsantwort zur Übergabe der Ausgabe der Lambda-Funktion an den Client als Antwort auf die `200 OK`-Methode einzurichten.

```
aws apigateway put-integration-response \
       --region us-west-2 \
       --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --status-code 200 \
       --selection-pattern ""
```

   Beim Festlegen des Auswahlmusters auf eine leere Zeichenfolge lautet die voreingestellte Antwort `200 OK`.

   Die erfolgreiche Antwort sollte nachstehender Antwort entsprechen:

```
 {
    "selectionPattern": "",
    "statusCode": "200"
}
```

8. Rufen Sie `create-deployment` auf, um die API für eine `test`-Stufe bereitzustellen:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

9. Testen Sie die API mithilfe des folgenden cURL-Befehls in einem Terminal:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=me' \
  -H 'authorization: AWS4-HMAC-SHA256 Credential={access_key}/20171020/us-
west-2/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=f327...5751'
```

Asynchronen Aufruf der Backend-Lambda-Funktion einrichten

Bei der nicht-proxy (benutzerdefinierten) Lambda-Integration wird die Backend-Lambda-Funktion standardmäßig synchron aufgerufen. Dies ist das gewünschte Verhalten für die meisten REST-API-Operationen. Einige Anwendungen erfordern jedoch eine asynchrone Durchführung (als

Batchvorgang oder Operation mit langer Latenzzeit), in der Regel durch eine separate Backend-Komponente. In diesem Fall wird die Lambda-Backend-Funktion asynchron aufgerufen, und die Frontend-REST-API-Methode gibt das Ergebnis nicht zurück.

Sie können die Lambda-Funktion für eine nicht-proxy Lambda-Integration so konfigurieren, dass sie asynchron aufgerufen wird, indem Sie `'Event'` als Lambda invocation type (Lambda-Aufruftyp) festlegen. Gehen Sie hierzu folgendermaßen vor:

Asynchronen Lambda-Aufruf in der API Gateway-Konsole konfigurieren

Damit alle Aufrufe asynchron sind:

- Fügen Sie in Integrationsanforderung einen `X-Amz-Invocation-Type`-Header mit dem statischen Wert `'Event'` hinzu.

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

1. Fügen Sie in Methodenanforderung einen `InvocationType`-Header hinzu.
2. Fügen Sie in Integrationsanforderung einen `X-Amz-Invocation-Type`-Header mit dem Mapping-Ausdruck `method.request.header.InvocationType` hinzu.
3. Clients können den `InvocationType: Event`-Header in API-Anforderungen für asynchrone Aufrufe oder `InvocationType: RequestResponse` für synchrone Aufrufe einschließen.

Asynchronen Lambda-Aufruf mit OpenAPI konfigurieren

Damit alle Aufrufe asynchron sind:

- Fügen Sie die `X-Amz-Invocation-Type` Kopfzeile zum x-amazon-apigateway-integrationAbschnitt hinzu.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
```

```
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" : "'Event'"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

1. Fügen Sie ein beliebiges OpenAPI Path Item Object dem folgenden Header hinzu.

```
"parameters" : [ {
"name" : "InvocationType",
"in" : "header",
"schema" : {
  "type" : "string"
}
} ]
```

2. Fügen Sie die `X-Amz-Invocation-Type` Kopfzeile zum x-amazon-apigateway-integrationAbschnitt hinzu.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" :
 "method.request.header.InvocationType"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

3. Clients können den `InvocationType: Event`-Header in API-Anforderungen für asynchrone Aufrufe oder `InvocationType: RequestResponse` für synchrone Aufrufe einschließen.

Konfigurieren Sie den asynchronen Lambda-Aufruf mit AWS CloudFormation

Die folgenden AWS CloudFormation Vorlagen zeigen, wie Sie den `AWS::ApiGateway::Method` für asynchrone Aufrufe konfigurieren.

Damit alle Aufrufe asynchron sind:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: AWS
        RequestParameters:
          integration.request.header.X-Amz-Invocation-Type: "'Event'"
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

Damit Clients entscheiden können, ob Aufrufe asynchron oder synchron sind:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      RequestParameters:
        method.request.header.InvocationType: false
      Integration:
        Type: AWS
        RequestParameters:
```

```
          integration.request.header.X-Amz-Invocation-Type:
 method.request.header.InvocationType
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

Clients können den `InvocationType: Event`-Header in API-Anforderungen für asynchrone Aufrufe oder `InvocationType: RequestResponse` für synchrone Aufrufe einschließen.

Lambda-Fehler in API Gateway behandeln

Bei benutzerdefinierten Lambda-Integrationen müssen Sie Fehler, die von Lambda in der Integrationsantwort zurückgegeben werden, für Ihre Kunden zu Standard-HTTP-Fehlerantworten zuordnen. Andernfalls werden Lambda-Fehler standardmäßig als `200 OK`-Meldungen zurückgegeben und ein solches Verhalten ist für die API-Benutzer nicht intuitiv.

Lambda kann zwei Fehlertypen zurückgeben, nämlich Standardfehler und benutzerdefinierte Fehler. In Ihrer API müssen Sie diese unterschiedlich handhaben.

Bei der Lambda-Proxy-Integration muss Lambda eine Ausgabe im folgenden Format zurückgeben:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}
```

Bei dieser Ausgabe entspricht `statusCode` einem `4XX`-Client-Fehler bzw. einem `5XX`-Server-Fehler. API Gateway behandelt diese Fehler über das Mapping des Lambda-Fehlers zu einer HTTP-Fehlerantwort gemäß den angegebenen `statusCode`. Damit API Gateway den Fehlertyp (z. B. `InvalidParameterException`) als Teil der Antwort an den Client weitergibt, muss die Lambda-Funktion einen Header (z. B. `"X-Amzn-ErrorType":"InvalidParameterException"`) in der `headers`-Eigenschaft enthalten.

Themen

- Standard-Lambda-Fehler in API Gateway behandeln
- Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln

Standard-Lambda-Fehler in API Gateway behandeln

Ein AWS Lambda-Standardfehler hat folgendes Format:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Hier ist `errorMessage` ein Zeichenfolgeausdruck des Fehlers. Der Wert `errorType` gibt eine sprachabhängige Fehler- oder Ausnahmeart an. `stackTrace` zeigt in einer Liste von Zeichenfolgeausdrücken das Stacktrace an, das zum Fehlerereignis geführt hat.

Nehmen wir beispielsweise die folgende JavaScript (Node.js) Lambda-Funktion.

```
export const handler = function(event, context, callback) {
    callback(new Error("Malformed input ..."));
};
```

Diese Funktion gibt den folgenden Lambda-Standardfehler mit der Fehlermeldung `Malformed input ...` zurück:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Bei der folgenden Python-Lambda-Funktion wird beispielsweise eine `Exception` mit derselben Fehlermeldung `Malformed input ...` ausgegeben:

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

Diese Funktion gibt den folgenden Standard-Lambda-Fehler zurück:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Beachten Sie, dass die Eigenschaftswerte `errorType` und `stackTrace` sprachabhängig sind. Der Standardfehler gilt auch für jedes Fehlerobjekt, das eine Erweiterung des `Error`-Objekts oder eine Subklasse der `Exception`-Klasse ist.

Um den Standard-Lambda-Fehler einer Methodenantwort zuzuordnen, müssen Sie zunächst einen HTTP-Statuscode für einen bestimmten Lambda-Fehler festlegen. Anschließend definieren Sie ein reguläres Ausdrucksmuster für die `selectionPattern`-Eigenschaft von IntegrationResponse, die dem angegebenen HTTP-Statuscode zugeordnet ist. In der API-Gateway-Konsole wird dieses `selectionPattern` im Abschnitt Integrationsantwort unter jeder Integrationsantwort als Lambda-Fehler-Regex ausgewiesen.

> **Note**
>
> API Gateway verwendet für das Antwort-Mapping Regexe-Anweisungen im Java-Pattern-Stil. Weitere Informationen finden Sie unter Pattern in der OracleDokumentation.

Wenn Sie beispielsweise einen neuen `selectionPattern`-Ausdruck mit der AWS CLI erstellen möchten, rufen Sie folgenden put-integration-response-Befehl auf:

```
aws apigateway put-integration-response --rest-api-id z0vprf0mdh --resource-id x3o5ih
 --http-method GET --status-code 400 --selection-pattern "Malformed.*" --region us-
west-2
```

Stellen Sie sicher, dass Sie auch den entsprechenden Fehlercode (`400`) für die Methodenantwort definieren. Andernfalls gibt API Gateway zur Laufzeit eine ungültige Konfigurationsfehler-Antwort aus.

> **Note**
>
> Zur Laufzeit ordnet API Gateway die `errorMessage` des Lambda-Fehlers dem Muster des regulären Ausdrucks für die `selectionPattern`-Eigenschaft zu. Wenn es eine Zuordnung gibt, gibt API Gateway den Lambda-Fehler als HTTP-Antwort des entsprechenden HTTP-Statuscodes zurück. Liegt keine Übereinstimmung vor, gibt API Gateway den Fehler als Standardmeldung zurück. Falls keine Standardmeldung konfiguriert ist, wird eine Fehlermeldung bezüglich einer ungültigen Konfiguration ausgegeben.
> Wenn als `selectionPattern`-Wert für eine bestimmte Antwort `.*` festgelegt wird, wird diese Antwort wieder als Standardantwort festgelegt. Der Grund hierfür ist, dass ein solches Auswahlmuster allen Fehlermeldungen entspricht, einschließlich Null, d. h. nicht spezifizierten Fehlermeldungen. Das daraus resultierende Mapping überschreibt das des Standard-Mapping.

Wenn Sie einen vorhandenen `selectionPattern`-Wert mithilfe der AWS CLI aktualisieren möchten, rufen Sie die Operation update-integration-response auf und ersetzen Sie den `/selectionPattern`-Pfadwert durch den angegebenen regulären Ausdruck des `Malformed*`-Musters.

Wenn Sie eine Integrationsantwort für einen bestimmten HTTP-Statuscode einrichten oder aktualisieren wollen, geben Sie den Ausdruck im Textfeld Lambda-Fehler-Regex ein, um den `selectionPattern`-Ausdruck über die Amazon API Gateway-Konsole zu definieren.

Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln

Anstatt einen Standardfehler wie oben beschriebenen auszugeben, können Sie in AWS Lambda ein benutzerdefiniertes Fehlerobjekt als JSON-Zeichenfolge zurückgeben. Der Fehler kann ein beliebiges gültiges JSON-Objekt sein. Beispielsweise gibt die folgende JavaScript (Node.js)-Lambda-Funktion einen benutzerdefinierten Fehler zurück:

```
export const handler = (event, context, callback) => {
```

```
    ...
    // Error caught here:
    var myErrorObj = {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
    callback(JSON.stringify(myErrorObj));
};
```

Das `myErrorObj`-Objekt muss in eine JSON-Zeichenfolge umgewandelt werden, bevor Sie `callback` zum Beenden der Funktion aufrufen. Andernfalls wird `myErrorObj` als `"[object Object]"`-Zeichenfolge zurückgegeben. Wenn eine Methode Ihrer API mit der vorhergehenden Lambda-Funktion integriert wird, erhält API Gateway eine Integrationsantwort mit dem folgenden Payload:

```
{
    "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
\"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
\"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
}
```

Wie bei jeder Integrationsantwort können Sie diese Fehlermeldung unverändert an die Methodenantwort übergeben. Alternativ können Sie die Nutzlast mithilfe einer Zuordnungsvorlage in ein anderes Format umwandeln. Nehmen Sie zum Beispiel die folgende Text-Mapping-Vorlage für eine Methodenantwort mit dem Statuscode `500`:

```
{
    errorMessage: $input.path('$.errorMessage');
}
```

Diese Vorlage wandelt den Integrationsantworttext mit der benutzerdefinierten JSON-Fehlerzeichenfolge in den folgenden Methodenantworttext um. Dieser Methodenantworttext enthält das benutzerdefinierte JSON-Fehlerobjekt:

```
{
```

```
    "errorMessage" : {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
};
```

Abhängig von Ihren API-Anforderungen müssen Sie möglicherweise einige oder alle benutzerdefinierte Fehlereigenschaften als Header-Parameter für die Methodenantwort übergeben. Dazu wenden Sie die benutzerdefinierten Fehler-Mappings aus dem Integrationsantworttext auf die Header der Methodenantwort an.

Beispielsweise definiert die folgende OpenAPI-Erweiterung das Mapping der Eigenschaften `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` und `errorMessage.trace` zu den Headern `error_type`, `error_status`, `error_trace_function` und `error_trace`:

```
"x-amazon-apigateway-integration": {
    "responses": {
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.error_trace_function":
 "integration.response.body.errorMessage.trace.function",
            "method.response.header.error_status":
 "integration.response.body.errorMessage.httpStatus",
            "method.response.header.error_type":
 "integration.response.body.errorMessage.errorType",
            "method.response.header.error_trace":
 "integration.response.body.errorMessage.trace"
          },
          ...
        }
    }
}
```

Zur Laufzeit deserialisiert API Gateway den Parameter `integration.response.body` bei der Ausführung von Header-Mappings. Diese Deserialisierung gilt nur für Text-zu-Header-Mappings bei benutzerdefinierten Lambda-Fehlermeldungen – sie gilt jedoch nicht für Text-zu-Text-Zuweisungen mit `$input.body`. Mit diesen Text-zu-Header-Mappings bei benutzerdefinierten Fehlermeldungen erhält der Client die folgenden Header als Teil der Methodenantwort (vorausgesetzt, die Header `error_status`, `error_trace`, `error_trace_function` und `error_type` wurden in der Methodenanforderung deklariert):

```
"error_status":"500",
"error_trace":"{\"function\":\"abc()\",\"line\":123,\"file\":\"abc.js\"}",
"error_trace_function":"abc()",
"error_type":"InternalServerError"
```

Die Eigenschaft `errorMessage.trace` des Integrationsantworttexts ist eine komplexe Eigenschaft. Sie wird dem Header `error_trace` als JSON-Zeichenfolge zugewiesen.

HTTP-Integrationen in API Gateway einrichten

Sie können eine API-Methode mit einem HTTP-Endpunkt mithilfe der HTTP-Proxy-Integration oder der benutzerdefinierten HTTP-Integration integrieren.

API Gateway unterstützt die folgenden Endpunkt-Ports: 80, 443 und 1024-65535.

Mit der Proxy-Integration ist die Einrichtung einfach. Sie müssen nur die HTTP-Methode und die HTTP-Endpunkt-URI entsprechend der Backend-Anforderungen festlegen, wenn Sie sich nicht mit der Codierung oder dem Caching von Inhalten befassen.

Mit der benutzerdefinierten Integration ist die Einrichtung komplexer. Zusätzlich zu den Einrichtungsschritten der Proxy-Integration müssen Sie angeben, wie die eingehenden Anforderungsdaten der Integrationsanforderung und die resultierenden Integrationsantwortdaten der Methodenantwort zugeordnet werden.

Themen

- HTTP-Proxy-Integrationen in API Gateway einrichten
- Benutzerdefinierte HTTP-API-Integrationen in API Gateway einrichten

HTTP-Proxy-Integrationen in API Gateway einrichten

Erstellen Sie zum Einrichten einer Proxy-Ressource mit dem HTTP-Proxy-Integrationstyp eine API-Ressource mit einem gierigen Pfadparameter (zum Beispiel `/parent/{proxy+}`) und integrieren Sie diese Ressource mit einem HTTP-Backend-Endpunkt (zum Beispiel `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) in die Methode `ANY`. Der gierige Pfadparameter muss am Ende des Ressourcenpfads stehen.

Wie bei einer Nicht-Proxy-Ressource können Sie eine Proxy-Ressource mit der HTTP-Proxy-Integration einrichten, indem Sie die API Gateway-Konsole verwenden, eine OpenAPI-Definitionsdatei importieren oder die API Gateway-REST-API direkt aufrufen. Detaillierte Anweisungen für die Verwendung der API Gateway-Konsole zur Konfiguration einer Proxy-Ressource mit der HTTP-API-Integration finden Sie unter Tutorial: REST-API mit HTTP-Proxy-Integration erstellen.

Die folgende OpenAPI-Definitionsdatei zeigt ein Beispiel für eine API mit einer Proxyressource, die in die PetStoreWebsite integriert ist.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T23:19:28Z",
      "title": "PetStoreWithProxyResource"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
```

```
                  "default": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.proxy": "method.request.path.proxy"
               },
               "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/
{proxy}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "ANY",
               "cacheNamespace": "rbftud",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "http_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://4z9giyi2c1.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
            }
         }
      }
   ]
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T23:19:28Z",
    "title": "PetStoreWithProxyResource"
  },
  "host": "4z9giyi2c1.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
```

```
    "https"
  ],
  "paths": {
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.proxy": "method.request.path.proxy"
          },
          "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/{proxy}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "ANY",
          "cacheNamespace": "rbftud",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "http_proxy"
        }
      }
    }
  }
}
```

In diesem Beispiel wird ein Cache-Schlüssel für den Pfadparameter `method.request.path.proxy` der Proxy-Ressource deklariert. Dies ist die Standardeinstellung,

wenn Sie die API mit der API Gateway-Konsole erstellen. Der Basispfad der API (`/test`entspricht einer Phase) ist der PetStore Seite der Website (`/petstore`) zugeordnet. Die einzelne Integrationsanfrage spiegelt die gesamte PetStore Website wider und verwendet dabei die Greedy Path-Variable der API und die ANY Catch-All-Methode. Die folgenden Beispiele veranschaulichen diese Spiegelung.

- Festlegen von **`ANY`** als **`GET`** und **`{proxy+}`** als **`pets`**

  Vom Frontend initiierte Methodenanforderung:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
  ```

  An das Backend gesendete Integrationsanforderung:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
  ```

  Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort `200 OK` mit der Nutzlast zurück, die den ersten Batch Haustiere enthält, wie vom Backend zurückgegeben.

- Festlegen von **`ANY`** als **`GET`** und **`{proxy+}`** als **`pets?type=dog`**

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
   HTTP/1.1
  ```

  An das Backend gesendete Integrationsanforderung:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
  ```

  Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort `200 OK` mit der Nutzlast zurück, die den ersten Batch der angegebenen Hunde enthält, wie vom Backend zurückgegeben.

- Festlegen von **`ANY`** als **`GET`** und **`{proxy+}`** als **`pets/{petId}`**

  Vom Frontend initiierte Methodenanforderung:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
  ```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die das angegebene Haustier enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **POST** und **{proxy+}** als **pets**

  Vom Frontend initiierte Methodenanforderung:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  An das Backend gesendete Integrationsanforderung:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Die Laufzeitinstanzen der Methode ANY und der Proxy-Ressource sind beide gültig. Der Aufruf gibt eine Antwort 200 OK mit der Nutzlast zurück, die das neu erstellte Haustier enthält, wie vom Backend zurückgegeben.

- Festlegen von **ANY** als **GET** und **{proxy+}** als **pets/cat**

  Vom Frontend initiierte Methodenanforderung:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

An das Backend gesendete Integrationsanforderung:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

Die Laufzeit-Instance des Proxy-Ressourcenpfads entspricht keinem Backend-Endpunkt und die resultierende Anforderung ist ungültig. Dies hat zur Folge, dass eine Antwort `400 Bad Request` mit der folgenden Fehlermeldung zurückgegeben wird.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    },
    {
      "key": "Pet2.price",
      "message": "Missing required field"
    }
  ]
}
```

- Festlegen von **`ANY`** als **`GET`** und **`{proxy+}`** als **`null`**

  Vom Frontend initiierte Methodenanforderung:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
  ```

  An das Backend gesendete Integrationsanforderung:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
  ```

  Die Zielressource ist der Proxy-Ressource übergeordnet, aber die Laufzeit-Instance der Methode `ANY` ist in der API auf dieser Ressource nicht definiert. Infolgedessen gibt diese `GET`-Anfrage eine `403 Forbidden`-Antwort mit der `Missing Authentication Token`-Fehlermeldung zurück, die von API Gateway zurückgegeben wird. Wenn die API die Methode `ANY` oder `GET` in der

übergeordneten Ressource (`/`) verfügbar macht, gibt der Aufruf eine Antwort `404 Not Found` mit der Meldung `Cannot GET /petstore` zurück, wie vom Backend zurückgegeben.

Wenn die URL des Zielendpunkts ungültig ist oder das HTTP-Verb gültig ist, aber nicht unterstützt wird, gibt das Backend für jede Clientanforderung die Antwort `404 Not Found` zurück. Bei einer nicht unterstützten HTTP-Methode wird eine Antwort `403 Forbidden` zurückgegeben.

Benutzerdefinierte HTTP-API-Integrationen in API Gateway einrichten

Mit der benutzerdefinierten HTTP-Integration haben Sie mehr Kontrolle darüber, welche Daten zwischen einer API-Methode und einer API-Integration übertragen werden sollen und wie Sie diese übertragen. Dies können Sie mithilfe von Daten-Mappings tun.

Als Teil der Einrichtung der Methodenanforderung legen Sie die requestParameters-Eigenschaft für eine Method-Ressource fest. Dies gibt an, welche Methodenanforderungsparameter, die von dem Client bereitgestellt werden, den Integrationsanforderungsparametern oder den geltenden Textkörpereigenschaften zugeordnet werden sollen, bevor sie an den Backend versendet werden. Anschließend legen Sie im Rahmen der Einrichtung der Integrationsanfrage die Eigenschaft requestParameters für die entsprechende Integrationsressource fest, um die parameter-to-parameter Zuordnungen anzugeben. Sie legen auch die requestTemplates-Eigenschaft fest, um die Mapping-Vorlagen anzugeben, eine für jeden unterstützten Inhaltstyp. Die Mapping-Vorlagen weisen Methodenanforderungsparameter oder Textkörper dem Anforderungstextkörper der Integration zu.

In ähnlicher Weise legen Sie im Rahmen der Einrichtung der Methodenantwort die Eigenschaft responseParameters für die Ressource fest. MethodResponse Dies gibt an, welche Methodenantwortparameter, die an den Client gesendet werden sollen, von den Integrationsantwortparametern oder bestimmten geltenden Textkörpereigenschaften zugeordnet werden sollen, die vom Backend zurückgegeben wurden. Anschließend legen Sie im Rahmen der Einrichtung der Integrationsantwort die ResponseParameters-Eigenschaft für die entsprechende IntegrationResponseRessource fest, um die Zuordnungen anzugeben. parameter-to-parameter Sie legen auch die responseTemplates-Zuordnung fest, um die Mapping-Vorlagen anzugeben, eine für jeden unterstützten Inhaltstyp. Die Mapping-Vorlagen weisen Methodenantwortparameter oder Textkörpereigenschaften der Integrationsantwort dem Antworttextkörper der Methode zu.

Weitere Hinweise zum Einrichten von Mapping-Vorlagen finden Sie unter Datentransformationen für REST-APIs einrichten.

Private API Gateway-Integrationen einrichten

Die private Integration von API Gateway macht die Bereitstellung Ihrer HTTP/HTTPS-Ressourcen hinter einer Amazon VPC für den Zugriff von Clients außerhalb des VPC einfach. Zum Erweitern des Zugriffs auf Ihre privaten VPC-Ressourcen über die VPC-Grenzen hinaus können Sie eine API mit privater Integration für offenen oder kontrollierten Zugriff erstellen. Sie können den Zugriff auf Ihre API steuern, indem Sie eine der von API Gateway unterstützten Autorisierungsmethoden verwenden.

Um eine private Integration zu erstellen, müssen Sie zunächst einen Network Load Balancer erstellen. Ihr Network Load Balancer muss über einen Listener verfügen, der Anfragen an Ressourcen in Ihrer VPC weiterleitet. Zur Verbesserung der Verfügbarkeit Ihrer API stellen Sie sicher, dass Ihr Network Load Balancer den Datenverkehr an Ressourcen in mehr als einer Availability Zone im AWS-Region weiterleitet. Anschließend erstellen Sie einen VPC-Link, mit dem Sie Ihre API und Ihren Network Load Balancer verbinden. Nachdem Sie einen VPC-Link erstellt haben, erstellen Sie private Integrationen, um den Datenverkehr von Ihrer API über Ihre VPC-Verbindung und Ihren Network Load Balancer an Ressourcen in Ihrer VPC zu leiten.

> **Note**
>
> Der Network Load Balancer und die API müssen demselben AWS Konto gehören.

Mit der privaten Amazon API Gateway-Integration können Sie den Zugriff auf HTTP/HTTPS-Ressourcen innerhalb einer VPC ohne detaillierte Kenntnisse der privaten Netzwerkkonfigurationen oder technologiespezifischer Appliances zulassen.

Themen

- Network Load Balancer für private API Gateway-Integrationen einrichten
- Erteilen von Berechtigungen zum Erstellen eines VPC-Links
- API Gateway-API mit privaten Integrationen unter Verwendung der API Gateway-Konsole einrichten
- Richten Sie eine API-Gateway-API mit privaten Integrationen ein, indem Sie den AWS CLI
- Einrichten einer API mit privaten Integrationen mit OpenAPI
- Für private Integrationen verwendete API Gateway-Konten

Network Load Balancer für private API Gateway-Integrationen einrichten

Die folgende Prozedur umreißt die Schritte zur Einrichtung eines Network Load Balancers (NLB) für private API Gateway-Integrationen unter Verwendung der Amazon EC2-Konsole und bietet Referenzen für detaillierte Anweisungen zu jedem Schritt.

Für jede VPC, in der Sie über Ressourcen verfügen, müssen Sie nur einen NLB und einen VPCLink konfigurieren. Der NLB unterstützt mehrere Listener und Zielgruppen pro NLB. Sie können jeden Service als einen bestimmten Listener auf dem NLB konfigurieren und einen einzelnen VPCLink zum Herstellen einer Verbindung mit dem NLB verwenden. Bei der Erstellung der privaten Integration in API Gateway definieren Sie jeden Service unter Verwendung des Ports, der dem Service zugewiesen ist. Weitere Informationen finden Sie unter the section called “Tutorial: Erstellen einer API mit privater Integration”.

> **Note**
>
> Der Network Load Balancer und die API müssen demselben AWS Konto gehören.

So erstellen Sie einen Network Load Balancer für die private Integration unter Verwendung der API Gateway-Konsole:

1. Melden Sie sich bei der Amazon EC2 EC2-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/ec2/.
2. Richten Sie einen Webserver in einer Amazon EC2-Instance ein. Ein Beispiel der Einrichtung finden Sie unter Installieren eines LAMP-Webservers auf Amazon Linux 2.
3. Erstellen Sie einen Network Load Balancer, registrieren Sie die EC2-Instance für eine Zielgruppe und fügen Sie die Zielgruppe einem Listener des Network Load Balancer hinzu. Für weitere Informationen befolgen Sie die Anweisungen unter Erste Schritte mit Network Load Balancern.
4. Nachdem der Network Load Balancer erstellt wurde, gehen Sie wie folgt vor:

    a. Notieren Sie sich den ARN des Network Load Balancers. Sie benötigen ihn, um einen VPC-Link in API Gateway für die Integration der API mit den VPC-Ressourcen hinter dem Network Load Balancer zu erstellen.

    b. Schalten Sie die Sicherheitsgruppenbewertung für PrivateLink aus.

    - Um die Sicherheitsgruppenauswertung für den PrivateLink Datenverkehr mithilfe der Konsole zu deaktivieren, können Sie auf der Registerkarte Sicherheit auf Bearbeiten

klicken. Deaktivieren Sie in den Sicherheitseinstellungen die Option Regeln für eingehenden PrivateLink Datenverkehr durchsetzen.

- Verwenden Sie den folgenden Befehl, um die Sicherheitsgruppenauswertung für den PrivateLink AWS CLI Datenverkehr mit dem zu deaktivieren:

```
aws elbv2 set-security-groups --load-balancer-
arn arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/net/
my-loadbalancer/abc12345 \
  --security-groups sg-123345a --enforce-security-group-inbound-rules-on-
private-link-traffic off
```

> **Note**
>
> Fügen Sie den API-Gateway-CIDRs keine Abhängigkeiten hinzu, da diese sich ohne vorherige Ankündigung ändern können.

Erteilen von Berechtigungen zum Erstellen eines VPC-Links

Damit Sie oder ein Benutzer in Ihrem Konto einen VPC-Link erstellen und pflegen können, müssen Sie bzw. der entsprechende Benutzer über Berechtigungen zum Erstellen, Löschen und Anzeigen der VPC-Endpunkt-Servicekonfigurationen, zum Ändern der VPC-Endpunkt-Service-Berechtigungen und zum Prüfen der Load Balancer verfügen. Führen Sie die folgenden Schritte aus, um diese Berechtigungen zu erteilen.

So erteilen Sie Berechtigungen zum Erstellen, Aktualisieren und Löschen eines VPC-Links

1. Erstellen Sie eine IAM-Richtlinie wie etwa folgende:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
```

```
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/vpclinks",
                "arn:aws:apigateway:us-east-1::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Erstellen oder wählen Sie eine IAM-Rolle aus und fügen Sie die vorhergehende Richtlinie der Rolle an.
3. Weisen Sie die IAM-Rolle Ihnen oder einem Benutzer in Ihrem Konto zu, der VPC-Links erstellt.

API Gateway-API mit privaten Integrationen unter Verwendung der API Gateway-Konsole einrichten

Anweisungen zur Einrichtung einer API mit privater Integration mithilfe der API Gateway-Konsole finden Sie unterTutorial: REST-API mit privater API Gateway-Integration erstellen.

Richten Sie eine API-Gateway-API mit privaten Integrationen ein, indem Sie den AWS CLI

Bevor Sie eine API mit der privaten Integration erstellen können, müssen Sie Ihre VPC-Ressource eingerichtet und einen Network Load Balancer erstellt und mit Ihrer VPC-Quelle als Ziel konfiguriert haben. Wenn die Anfragen nicht erfüllt sind, folgen Sie Network Load Balancer für private API Gateway-Integrationen einrichten, um die VPC-Ressource zu installieren, einen Network Load Balancer zu erstellen und die VPC-Ressource als Ziel des Network Load Balancers festzulegen.

> **Note**
>
> Der Network Load Balancer und die API müssen demselben AWS Konto gehören.

Damit Sie eine `VpcLink` erstellen und verwalten können, müssen Sie auch die entsprechenden Berechtigungen konfiguriert haben. Weitere Informationen finden Sie unter Erteilen von Berechtigungen zum Erstellen eines VPC-Links.

> **Note**
>
> Sie benötigen nur die Berechtigungen zum Erstellen eines `VpcLink` in Ihrer API. Sie benötigen nicht die Berechtigungen zum Verwenden des `VpcLink`.

Nachdem der Network Load Balancer erstellt wurde, notieren Sie sich den zugehörigen ARN. Sie benötigen ihn zum Erstellen eines VPC-Links für die private Integration.

Um eine API mit der privaten Integration einzurichten, verwenden Sie AWS CLI

1. Erstellen Sie einen `VpcLink` mit Ausrichtung auf den angegebenen Network Load Balancer.

   ```
   aws apigateway create-vpc-link \
       --name my-test-vpc-link \
       --target-arns arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/
   net/my-vpclink-test-nlb/1234567890abcdef
   ```

   Die Ausgabe dieses Befehls bestätigt den Eingang der Anfrage und zeigt den `PENDING`-Status des soeben erstellten `VpcLink` an.

   ```
   {
       "status": "PENDING",
       "targetArns": [
           "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
   vpclink-test-nlb/1234567890abcdef"
       ],
       "id": "gim7c3",
       "name": "my-test-vpc-link"
   }
   ```

Es dauert 2-4 Minuten, bis API Gateway die Erstellung des `VpcLink` abgeschlossen hat. Wenn der Vorgang erfolgreich abgeschlossen wird, lautet der `status` `AVAILABLE`. Sie können dies überprüfen, indem Sie den folgenden CLI-Befehl aufrufen:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Wenn der Vorgang fehlschlägt, erhalten Sie den Status `FAILED` und die `statusMessage` mit der Fehlermeldung. Wenn Sie beispielsweise versuchen, `VpcLink` mit einem Network Load Balancer zu erstellen, der bereits einem VPC-Endpunkt zugewiesen ist, erhalten Sie Folgendes in der Eigenschaft `statusMessage`:

```
"NLB is already associated with another VPC Endpoint Service"
```

Erst nachdem der `VpcLink` erfolgreich erstellt wurde, können wir die API erstellen und über den `VpcLink` in die VPC-Ressource integrieren.

Notieren Sie sich den Wert `id` im neu erstellten `VpcLink` (*`gim7c3`* in der vorangegangenen Ausgabe). Sie benötigen ihn zum Einrichten der privaten Integration.

2. Richten Sie eine API ein, indem Sie eine API Gateway-`RestApi`-Ressource erstellen:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Notieren Sie sich den in der Antwort angegebenen `RestApi`-Wert für die `id`. Sie benötigen diesen Wert später, um weitere Vorgänge in der API, auszuführen.

Zur Veranschaulichung erstellen wir eine API mit nur einer `GET`-Methode in der Root-Ressource (`/`) und integrieren die Methode im `VpcLink`.

3. Richten Sie die `GET /`-Methode ein. Rufen Sie zunächst die Kennung der Root-Ressource () (`/`):

```
aws apigateway get-resources --rest-api-id  abcdef123
```

Notieren Sie sich den `id`-Wert des Pfads `/` in der Ausgabe. In diesem Beispiel nehmen wir an, er lautet *`skpp60rab7`*.

Richten Sie die Methodenanforderung für die API-Methode ei `GET /`:

```
aws apigateway put-method \
```

```
        --rest-api-id  abcdef123 \
        --resource-id skpp60rab7 \
        --http-method GET \
        --authorization-type "NONE"
```

Wenn Sie die Proxy-Integration mit dem `VpcLink` nicht verwenden, müssen Sie mindestens eine Methodenantwort des Statuscodes `200` einrichten. In diesem Beispiel verwenden wir die Proxy-Integration.

4. Richten Sie die private Integration des Typs `HTTP_PROXY` ein und rufen Sie den Befehl `put-integration` wie folgt auf:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id gim7c3
```

Für eine private Integration müssen Sie `connection-type` auf „`VPC_LINK`“ und `connection-id` entweder auf die Kennung Ihres `VpcLink` oder auf eine Stufenvariable festlegen, die auf Ihre `VpcLink`-ID verweist. Der Parameter `uri` wird nicht für das Routing von Anforderungen an Ihren Endpunkt verwendet, wohl aber zum Festlegen des `Host`-Headers und für die Zertifikatsvalidierung.

Der Befehl gibt die folgende Ausgabe zurück:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "connectionId": "gim7c3",
    "uri": "http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com",
    "connectionType": "VPC_LINK",
    "httpMethod": "GET",
    "cacheNamespace": "skpp60rab7",
    "type": "HTTP_PROXY",
    "cacheKeyParameters": []
}
```

Legen Sie unter Verwendung einer Stufenvariable die Eigenschaft `connectionId` beim Erstellen der Integration fest:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkId}"
```

Stellen Sie sicher, dass Sie den Stufenvariablenausdruck in doppelte Anführungszeichen (`${stageVariables.vpcLinkId}`) setzen und das Zeichen $ mit einem Escape-Zeichen versehen.

Alternativ können Sie die Integration aktualisieren, um den Wert `connectionId` mit einer Stufenvariable zurückzusetzen:

```
 aws apigateway update-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --http-method GET \
    --patch-operations '[{"op":"replace","path":"/
connectionId","value":"${stageVariables.vpcLinkId}"}]'
```

Verwenden Sie unbedingt eine in Text umgewandelte JSON-Liste als `patch-operations`-Parameterwert.

Sie können eine Stufenvariable verwenden, um Ihre API in eine andere VPC oder einen anderen Network Load Balancer zu integrieren, indem Sie den Wert der `VpcLink`-Stufenvariablen zurücksetzen.

Da wir die private Proxy-Integration verwendet haben, ist die API jetzt für die Bereitstellung und für Testläufe bereit. Mit der Nicht-Proxy-Integration müssen Sie auch die Methoden- und Integrationsantwort einrichten ähnlich wie beim Einrichten einer API mit benutzerdefinierten HTTP-Integrationen.

5. Stellen Sie die API bereit, um sie zu testen. Dies ist erforderlich, wenn Sie die Stufenvariable als Platzhalter der `VpcLink`-ID verwendet haben. Zum Bereitstellen der API mit einer Stufenvariablen rufen Sie den Befehl `create-deployment` wie folgt auf:

   ```
   aws apigateway create-deployment \
       --rest-api-id abcdef123 \
       --stage-name test \
       --variables vpcLinkId=gim7c3
   ```

   Zum Aktualisieren der Stufenvariablen mit einer anderen `VpcLink`-ID (z. B. *asf9d7*) rufen Sie den Befehl `update-stage` auf:

   ```
   aws apigateway update-stage \
       --rest-api-id abcdef123 \
       --stage-name test \
       --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
   ```

   Rufen Sie Ihre API mit dem folgenden Befehl auf:

   ```
   curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
   ```

   Alternativ können Sie die invoke-URL der API in einem Webbrowser eingeben, um das Ergebnis anzuzeigen.

   Wenn Sie die Eigenschaft `connection-id` mit dem `VpcLink`-ID-Literal hartcodieren, können Sie auch `test-invoke-method` verwenden, um das Aufrufen der API vor der Bereitstellung zu testen.

Einrichten einer API mit privaten Integrationen mit OpenAPI

Sie können eine API mit der privaten Integration durch Importieren der API-OpenAPI-Datei einrichten. Die Einstellungen werden ähnlich wie die OpenAPI-Definitionen einer API mit HTTP-Integrationen festgelegt.Dabei gelten jedoch die folgenden Ausnahmen:

- Sie müssen `connectionType` explizit auf `VPC_LINK` festlegen.
- Sie müssen `connectionId` explizit auf die ID eines `VpcLink` oder auf eine Stufenvariable festlegen, die auf die ID eines `VpcLink` verweist.

- Der Parameter `uri` in der privaten Integration weist auf einen HTTP/HTTPS-Endpunkt in der VPC, wird jedoch stattdessen zum Einrichten des `Host`-Headers der Integrationsanforderung verwendet.
- Der Parameter `uri` in der privaten Integration mit einem HTTPS-Endpunkt in der VPC wird verwendet, um den angegebenen Domänennamen anhand des Namens in dem auf dem VPC-Endpunkt installierten Zertifikats zu überprüfen.

Sie können eine Stufenvariable als Referenz auf die `VpcLink`-ID verwenden. Sie können den ID-Wert auch `connectionId` direkt zuweisen.

Die folgende JSON-formatierte API-OpenAPI-Datei zeigt ein Beispiel für eine API mit einem VPC-Link, auf den eine Stufenvariable (`${stageVariables.vpcLinkId}`) verweist:

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLink"
  },
  "host": "p3wocvip9a.execute-api.us-west-2.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
```

```
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "http://my-vpclink-test-nlb-1234567890abcdef.us-
east-2.amazonaws.com",
          "passthroughBehavior": "when_no_match",
          "connectionType": "VPC_LINK",
          "connectionId": "${stageVariables.vpcLinkId}",
          "httpMethod": "GET",
          "type": "http_proxy"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Für private Integrationen verwendete API Gateway-Konten

Die folgenden regionsspezifischen API Gateway-Konto-IDs werden Ihrem VPC-Endpunkt-Service automatisch als `AllowedPrincipals` hinzugefügt, wenn Sie einen `VpcLink` erstellen.

Region	Konto-ID
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099

Region	Konto-ID
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Mock-Integrationen in API Gateway einrichten

Amazon API Gateway unterstützt die Mock-Integration von API-Methoden. Diese Funktion ermöglicht es API-Entwicklern, API-Antworten direkt von API Gateway zu generieren, ohne ein Integrations-Backend zu benötigen. Als API-Developer können Sie diese Funktion nutzen, um abhängige Teams zu entsperren, die noch vor Beendigung der Projektentwicklung mit der API arbeiten müssen. Zudem lässt sich mit dieser Funktion eine Startseite für die API bereitstellen, die eine Übersicht über die API und die Navigation zu dieser bietet. Ein Beispiel für eine solche Startseite finden Sie in der Integrationsanforderung und -antwort der GET-Methode für die Stammressource der Beispiel-API unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Als API-Entwickler entscheiden Sie, wie API Gateway auf eine Mock-Integrationsanfrage antwortet. Dazu konfigurieren Sie die Integrationsanforderung und -antwort der Methode, um einer Antwort einen bestimmten Statuscode zuzuweisen. Damit eine Methode mit Mock-Integration eine 200-Antwort zurückgibt, konfigurieren Sie die Text-Mapping-Vorlage der Integrationsanforderung so, dass Folgendes zurückgegeben wird.

```
{"statusCode": 200}
```

Konfigurieren Sie eine 200-Integrationsantwort für folgende Text-Mapping-Vorlage, zum Beispiel:

```
{
    "statusCode": 200,
    "message": "Go ahead without me."
}
```

Damit die Methode beispielsweise eine 500-Fehlerantwort zurückgibt, richten Sie dem entsprechend die Text-Mapping-Vorlage der Integrationsanforderung so ein, dass Folgendes zurückgegeben wird.

```
{"statusCode": 500}
```

Richten Sie eine 500-Integrationsantwort beispielsweise mit der folgenden Mapping-Vorlage ein:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

Alternativ können Sie eine Methode der Mock-Integration die standardmäßige Integrationsantwort zurückgeben lassen, ohne die Mapping-Vorlage der Integrationsanforderung zu definieren. Die standardmäßige Integrationsantwort ist die mit einem nicht definierten HTTP status regex. Stellen Sie sicher, dass geeignete Pass-Through-Verhaltensweisen festgelegt wurden.

> **Note**
>
> Mock-Integrationen sind nicht für die Unterstützung großer Antwortvorlagen vorgesehen. Wenn Sie sie für Ihren Anwendungsfall benötigen, sollten Sie stattdessen die Verwendung einer Lambda-Integration in Betracht ziehen.

Mit der Mapping-Vorlage einer Integrationsanforderung können Sie Anwendungslogik einbringen, um basierend auf bestimmten Bedingungen zu entscheiden, welche Mock-Integrationsantwort zurückgegeben wird. Sie können bei der eingehenden Anforderung beispielsweise einen `scope`-Abfrageparameter verwenden, um zu bestimmen, ob eine Erfolgs- oder Fehlerantwort zurückgegeben wird:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

Auf diese Weise lässt die Methode der Mock-Integration interne Aufrufe durch und lehnt andere Aufruftypen mit einer Fehlermeldung ab.

In diesem Abschnitt wird beschrieben, wie die API Gateway-Konsole verwendet wird, um die Mock-Integration für eine API-Methode zu ermöglichen.

Themen

- Mock-Integration mit der API Gateway-Konsole aktivieren

Mock-Integration mit der API Gateway-Konsole aktivieren

Hierfür muss in API Gateway eine Methode verfügbar sein. Folgen Sie den Anweisungen in Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen.

1. Wählen Sie eine API-Ressource aus und klicken Sie auf Methode erstellen.

   Die Methode richten Sie wie folgt ein:

   a. Für Methodentyp wählen Sie eine HTTP-Methode aus.
   b. Für den Integrationstyp wählen Sie Mock aus.
   c. Wählen Sie Methode erstellen aus.
   d. Klicken Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen auf Bearbeiten.
   e. Klicken Sie auf Parameter für URL-Abfragezeichenfolgen. Klicken Sie auf Abfragezeichenfolge hinzufügen und geben Sie **`scope`** als Namen ein. Dieser Abfrageparameter stellt fest, ob es sich um einen internen Aufrufer handelt.
   f. Wählen Sie Speichern.
2. Klicken Sie auf der Registerkarte Methodenantwort aufAntwort erstellen und gehen Sie dann wie folgt vor:

   a. Für HTTP-Status geben Sie **`500`** ein.
   b. Wählen Sie Speichern.
3. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
4. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:

   a. Wählen Sie Add mapping template.
   b. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
   c. Geben Sie für Vorlagentext Folgendes ein:

   ```
   {
     #if( $input.params('scope') == "internal" )
       "statusCode": 200
     #else
       "statusCode": 500
     #end
   }
   ```

   d. Wählen Sie Speichern.
5. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
6. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:

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

b. Geben Sie für Vorlagentext Folgendes ein:

```
{
    "statusCode": 200,
    "message": "Go ahead without me"
}
```

c. Wählen Sie Speichern.

7. Wählen Sie Create response (Antwort erstellen) aus.

   Gehen Sie wie folgt vor, um eine Antwort 500 zu erstellen:

   a. Machen Sie für HTTP status regex (HTTP-Status-RegEx) den Eintrag **`5\d{2}`**.

   b. Wählen Sie für Status der Methodenantwort **`500`** aus.

   c. Wählen Sie Speichern.

   d. Klicken Sie unter 5\d{2} - Antwort auf Bearbeiten.

   e. Wählen Sie Zuordnungsvorlagen aus und klicken Sie dann auf Zuordnungsvorlage hinzufügen.

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

   g. Geben Sie für Vorlagentext Folgendes ein:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

   h. Wählen Sie Speichern.

8. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen. Gehen Sie wie folgt vor, um Ihre Mock-Integration zu testen:

   a. Geben Sie unter Abfragezeichenfolgen `scope=internal` ein. Wählen Sie Test aus. Das Testergebnis zeigt Folgendes:

```
Request: /?scope=internal
Status: 200
Latency: 26 ms
```

```
Response Body

{
  "statusCode": 200,
  "message": "Go ahead without me"
}

Response Headers

{"Content-Type":"application/json"}
```

b. Geben Sie `scope=public` unter `Query strings` ein oder lassen Sie das Feld leer. Wählen Sie Test aus. Das Testergebnis zeigt Folgendes:

```
Request: /
Status: 500
Latency: 16 ms
Response Body

{
  "statusCode": 500,
  "message": "The invoked method is not supported on the API resource."
}

Response Headers

{"Content-Type":"application/json"}
```

Sie können auch Header in einer Mock-Integrationsantwort zurückgeben, indem Sie der Methodenantwort zunächst einen Header hinzufügen und anschließend ein Header-Mapping in der Integrationsantwort einrichten. Die API Gateway-Konsole setzt die CORS-Unterstützung über die Rückgabe von CORS-fähigen Headern um.

Verwenden der Anforderungsvalidierung in API Gateway

Sie können API Gateway so konfigurieren, dass es eine grundlegende Validierung einer API-Anfrage durchführt, bevor es mit der Integrationsanfrage fortfährt. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage sofort fehl, gibt eine 400-Fehlerantwort an den Aufrufer zurück

und veröffentlicht die Überprüfungsergebnisse in CloudWatch Logs. Damit werden unnötige Aufrufe im Backend vermieden. Vor allem aber können Sie die Validierung so speziell auf die Anwendung ausrichten. Sie können einen Anforderungstext validieren, indem Sie überprüfen, ob die erforderlichen Anforderungsparameter gültig sind und nicht den Wert Null haben, oder indem Sie für komplexere Datenvalidierungen ein Modellschema angeben.

Themen

Überblick über die grundlegende Anforderungsvalidierung in API Gateway

API Gateway kann die grundlegende Anforderungsvalidierung durchführen, sodass Sie sich auf die anwendungsspezifische Validierung im Backend konzentrieren können. Für die grundlegende Validierung überprüft API Gateway eine oder beide der folgenden Bedingungen:

- Die erforderlichen Anforderungsparameter in der URI, der Abfragezeichenfolge und dem Header einer eingehenden Anforderung sind vorhanden und nicht leer.
- Die entsprechende Anforderungsnutzlast entspricht dem konfigurierten JSON-Schema-Anforderungsmodell der Methode.

Zum Aktivieren der grundlegenden Validierung legen Sie Validierungsregeln für eine Anforderungsvalidierung fest, fügen diese dem Schema der Anforderungsvalidierungen für die API hinzu und ordnen anschließend die Validierung den einzelnen API-Methoden zu.

> **Note**
>
> Das Anfordern einer Textüberprüfung an und Integrations-Pass-Through-Verhalten sind zwei separate Themen. Wenn die Nutzlast einer Anforderung kein passendes Modellschema hat, können Sie Pass-Through wählen oder die ursprüngliche Nutzlast blockieren. Weitere Informationen finden Sie unter Integrations-Pass-Through-Verhalten.

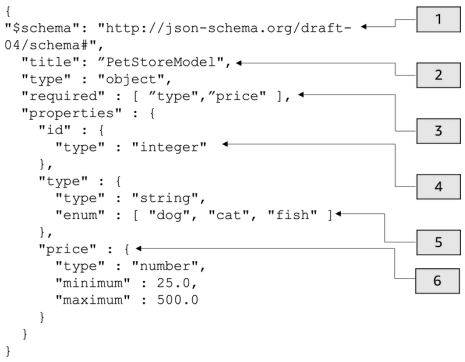
Datenmodelle

In API Gateway definiert ein Modell die Datenstruktur eines Payloads. In API Gateway werden Modelle unter Verwendung des JSON-Schemaentwurfs 4 definiert. Das folgende JSON-Objekt enthält Beispieldaten im Tierhandlungsbeispiel.

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

Die Daten enthalten die `id`, den `type` und den `price` des Haustiers. Ein Modell dieser Daten ermöglicht Ihnen Folgendes:

- Verwenden der grundlegenden Anforderungsvalidierung.
- Erstellen von Zuweisungsvorlagen für die Datentransformation.
- Erstellen eines benutzerdefinierten Datentyps (UDT), wenn Sie ein SDK generieren.



In diesem Modell gilt:

1. Das Objekt `$schema` stellt eine gültige JSON-Schema-Versionskennung dar. Dieses Schema ist der JSON-Schemaentwurf v4.
2. Das Objekt `title` ist eine lesbare Kennung des Modells. Dieser Titel ist `PetStoreModel`.
3. Das `required`-Validierungsschlüsselwort erfordert `type` und `price` für die grundlegende Anforderungsvalidierung.
4. Die `properties` des Modells sind `id`, `type` und `price`. Jedes Objekt hat Eigenschaften, die im Modell beschrieben werden.

5. Das Objekt `type` kann nur die Werte `dog`, `cat` oder `fish` haben.
6. Das Objekt `price` ist eine Zahl und auf einen `minimum`-Wert von 25 und einen `maximum`-Wert von 500 beschränkt.

PetStore Modell

```
1 {
2 "$schema": "http://json-schema.org/draft-04/schema#",
3  "title": "PetStoreModel",
4  "type" : "object",
5  "required" : [ "price", "type" ],
6  "properties" : {
7    "id" : {
8      "type" : "integer"
9    },
10    "type" : {
11      "type" : "string",
12      "enum" : [ "dog", "cat", "fish" ]
13    },
14    "price" : {
15      "type" : "number",
16      "minimum" : 25.0,
17      "maximum" : 500.0
18    }
19  }
20 }
```

In diesem Modell gilt:

1. In Zeile 2 stellt das Objekt `$schema` eine gültige JSON-Schema-Versionskennung dar. Dieses Schema ist der JSON-Schemaentwurf v4.
2. In Zeile 3 ist das Objekt `title` eine lesbare Kennung des Modells. Dieser Titel ist `PetStoreModel`.
3. In Zeile 5 erfordert das `required` Validierungsschlüsselwort `type` und `price` für die grundlegende Anforderungsvalidierung.
4. In den Zeilen 6 bis 17 sind die `properties` des Modells `id` ,`type` und `price`. Jedes Objekt hat Eigenschaften, die im Modell beschrieben werden.
5. In Zeile 12 `type` kann das Objekt nur die Werte `dog`, `cat` oder `fish` haben.

6. In den Zeilen 14 bis 17 ist das Objekt `price` eine Zahl und durch einen `minimum`-Wert von 25 und einen `maximum`-Wert von 500 eingeschränkt.

Erstellen komplexerer Modelle

Sie können das `$ref`-Primitiv verwenden, um wiederverwendbare Definitionen für längere Modelle zu erstellen. Sie können beispielsweise eine Definition mit der Bezeichnung `Price` im `definitions`-Abschnitt zur Beschreibung des `price`-Objekts erstellen. Der Wert von `$ref` ist die `Price`-Definition.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
      "Price": {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
            }
      }
}
```

Sie können auch auf ein anderes Modellschema verweisen, das in einer externen Modelldatei definiert ist. Stellen Sie den Wert der `$ref`-Eigenschaft auf die Position des Modells ein. Im folgenden Beispiel ist das `Price`-Modell im `PetStorePrice`-Modell in der API `a1234` definiert.

```
{
```

```
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

Das längere Modell kann auf das `PetStorePrice`-Modell verweisen.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Verwendung von Ausgabedatenmodellen

Wenn Sie Ihre Daten umwandeln, können Sie in der Integrationsantwort ein Nutzlastmodell definieren. Ein Nutzlastmodell kann verwendet werden, wenn Sie ein SDK generieren. Bei stark typisierten Sprachen wie Java, Objective-C oder Swift entspricht das Objekt einem benutzerdefinierten Datentyp (UDT). API Gateway erstellt eine UDT, wenn Sie ihm bei der Generierung eines SDK ein Datenmodell zur Verfügung stellen. Weitere Informationen zu Datentransformationen finden Sie unter Grundlegendes zu Zuweisungsvorlagen.

Ausgabedaten	

	<pre>{
	[
	{</pre>

<table>
<tr><td></td><td><pre>
    "description" : "Item 1 is a
 dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a
 cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a
 fish.",
    "askingPrice" : 0.99
  }
]
}
</pre></td></tr>
<tr><td>Ausgabemodell</td><td><pre>
{
"$schema": "http://json-schema.org/
draft-04/schema#",
  "title": ”PetStoreOutputModel",
  "type" : "object",
  "required" : ["description",
 "askingPrice"],
  "properties" : {
    "description" : {
      "type" : "string"
    },
    "askingPrice" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
</pre></td></tr>
</table>

Mit diesem Modell können Sie ein SDK aufrufen, um die Eigenschaftswerte `description` und `askingPrice` durch Lesen der Eigenschaften `PetStoreOutputModel[i].description` und `PetStoreOutputModel[i].askingPr`

Wenn kein Modell zur Verfügung gestellt wird, verwendet API Gateway das leere Modell zur Erstellung einer Standard-UDT.

Nächste Schritte

- In diesem Abschnitt finden Sie Ressourcen, die Sie verwenden können, um mehr über die in diesem Thema vorgestellten Konzepte zu erfahren.

  Sie können den Tutorials zur Validierung von Anfragen folgen:

  - Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole
  - Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI
  - Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition
- Weitere Informationen zu Datenumwandlungs- und Zuweisungsvorlagen finden Sie unter Grundlegendes zu Zuweisungsvorlagen.
- Sie können auch kompliziertere Datenmodelle sehen. Siehe Beispieldatenmodelle und Zuordnungsvorlagen für API Gateway.

Grundlegende Anforderungsvalidierung in API Gateway einrichten

In diesem Abschnitt wird gezeigt, wie die Anforderungsvalidierung für API Gateway mithilfe der Konsole und einer OpenAPI-Definition eingerichtet wird. AWS CLI

Themen

- Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole
- Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI
- Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition

Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole

Sie können die API-Gateway-Konsole verwenden, um eine Anfrage zu validieren, indem Sie einen von drei Validatoren für eine API-Anfrage auswählen:

- Text validieren.
- Abfragezeichenfolgeparameter und Header validieren.
- Text, Abfragezeichenfolgeparameter und Header validieren.

Wenn Sie einen der Validatoren auf eine API-Methode anwenden, fügt die API-Gateway-Konsole den Validator zur API-Map hinzu. RequestValidators

Um diesem Tutorial zu folgen, verwenden Sie eine AWS CloudFormation Vorlage, um eine unvollständige API Gateway zu erstellen. Diese API hat eine `/validator`-Ressource mit den Methoden `GET` und `POST`. Beide Methoden sind in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt integriert. Sie werden zwei Arten der Anforderungsvalidierung konfigurieren:

- In der `GET`-Methode konfigurieren Sie die Anforderungsvalidierung für URL-Abfragezeichenfolgen-Parameter.
- In der `POST`-Methode konfigurieren Sie die Anforderungsvalidierung für den Anforderungstext.

Dadurch können nur bestimmte API-Aufrufe an die API weitergeleitet werden.

Laden Sie die Vorlage zur App-Erstellung für AWS CloudFormation herunter und entpacken Sie sie. Sie verwenden diese Vorlage, um eine unvollständige API zu erstellen. Sie schließen die restlichen Schritte in der API-Gateway-Konsole ab.

Um einen Stack zu erstellen AWS CloudFormation

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

AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres AWS CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

So wählen Sie Ihre neu erstellte API aus

1. Wählen Sie den neu erstellten **`request-validation-tutorial-console`**-Stack aus.
2. Wählen Sie Resources aus.
3. Wählen Sie unter Physische ID Ihre API aus. Dieser Link leitet Sie zur API-Gateway-Konsole weiter.

Bevor Sie die Methoden `GET` und `POST` ändern, müssen Sie ein Modell erstellen.

So erstellen Sie ein Modell

1. Ein Modell ist erforderlich, um die Anforderungsvalidierung für den Text einer eingehenden Anfrage zu verwenden. Klicken Sie im Hauptnavigationsbereich auf Modelle, um ein Modell zu erstellen.
2. Wählen Sie Modell erstellen aus.
3. Geben Sie unter Name **`PetStoreModel`** ein.
4. Geben Sie für Inhaltstyp **`application/json`** ein. Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Geben Sie **`$default`** ein, um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden.
5. Geben Sie für Beschreibung **`My PetStore Model`** als die Modellbeschreibung ein.
6. Für Modellschema kopieren Sie das folgende Modell in den Code-Editor ein und klicken dann auf Modell erstellen.

```
{
  "type" : "object",
  "required" : [ "name", "price", "type" ],
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
```

```
    "name" : {
      "type" : "string"
    },
    "price" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Weitere Informationen zum Modell finden Sie unter Datenmodelle.

Konfigurieren einer Anforderungsvalidierung für eine **`GET`**-Methode

1. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die GET-Methode in der Ressourcenstruktur aus.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Wählen Sie für Anforderungs-Validierer die Option Abfragezeichenfolgeparameter und -Header validieren aus.
4. Wählen Sie URL-Abfragezeichenfolgen-Parameter aus und gehen Sie dann wie folgt vor:
   a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
   b. Geben Sie unter Name **`petType`** ein.
   c. Aktivieren Sie die Option Erforderlich.
   d. Caching bleibt ausgeschaltet.
5. Wählen Sie Speichern.
6. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
7. Wählen Sie URL-Abfragezeichenfolgen-Parameter aus und gehen Sie dann wie folgt vor:
   a. Wählen Sie Abfragezeichenfolge hinzufügen aus.
   b. Geben Sie unter Name **`petType`** ein.
   c. Geben Sie für Zugeordnet von **`method.request.querystring.petType`** ein. Dadurch wird **`petType`** dem Typ des Haustiers zugeordnet.

Weitere Informationen zur Datenzuordnung finden Sie im Tutorial zur Datenzuordnung.

d. Caching bleibt ausgeschaltet.

8. Wählen Sie Speichern.

So testen Sie die Anforderungsvalidierung für die **GET**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Abfragezeichenfolgen geben Sie **petType=dog** ein und klicken dann auf Test.
3. Der Methodentest gibt `200 OK` und eine Liste von Hunden zurück.

   Informationen zur Umwandlung dieser Ausgabedaten finden Sie im Tutorial zur Datenzuordnung.
4. Entfernen Sie **petType=dog** und wählen Sie Test aus.
5. Der Methodentest gibt einen `400`-Fehler mit der folgenden Fehlermeldung zurück:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Konfigurieren der Anforderungsvalidierung für die **POST**-Methode

1. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die POST-Methode aus.
2. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.
3. Für Anforderungs-Validator wählen Sie die Option Text validieren aus.
4. Klicken Sie unter Anforderungstext auf Modell hinzufügen.
5. Geben Sie **application/json** als Inhaltstyp den Wert ein und wählen Sie dann für Modell aus. PetStoreModel
6. Wählen Sie Speichern.

Testen der Anforderungsvalidierung für eine **POST**-Methode

1. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Für Anforderungstext kopieren Sie Folgendes in den Code-Editor:

   ```
   {
     "id": 2,
     "name": "Bella",
     "type": "dog",
     "price": 400
   }
   ```

   Wählen Sie Test aus.
3. Der Methodentest gibt `200 OK` und eine Erfolgsmeldung zurück.
4. Für Anforderungstext kopieren Sie Folgendes in den Code-Editor:

   ```
   {
     "id": 2,
     "name": "Bella",
     "type": "dog",
     "price": 4000
   }
   ```

   Wählen Sie Test aus.
5. Der Methodentest gibt einen `400`-Fehler mit der folgenden Fehlermeldung zurück:

   ```
   {
    "message": "Invalid request body"
   }
   ```

   Der Grund für den ungültigen Anforderungstext wird ganz unten in den Testprotokollen angegeben. In diesem Fall lag der Preis des Haustiers außerhalb des im Modell angegebenen Höchstpreises.

Um einen AWS CloudFormation Stapel zu löschen

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

Nächste Schritte

- Informationen zur Transformation von Ausgabedaten und zur Durchführung weiterer Datenzuordnungen finden Sie im Tutorial zur Datenzuordnung.
- Folgen Sie dem Abschnitt Einrichtung einer einfachen Anforderungsvalidierung mithilfe der AWS CLI, um ähnliche Schritte mit der AWS CLI durchzuführen.

Richten Sie die grundlegende Anforderungsvalidierung mit dem ein AWS CLI

Sie können einen Validator erstellen, um die Anforderungsvalidierung mithilfe der AWS CLI einzurichten. Um diesem Tutorial zu folgen, verwenden Sie eine AWS CloudFormation Vorlage, um eine unvollständige API Gateway zu erstellen.

> **Note**
>
> Dies ist nicht dieselbe AWS CloudFormation Vorlage wie das Konsolen-Tutorial.

Mithilfe einer vorab bereitgestellten `/validator`-Ressource erstellen Sie die Methoden `GET` und `POST`. Beide Methoden werden in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt integriert. Sie konfigurieren die folgenden zwei Anforderungsvalidierungen:

- Für die `GET`-Methode erstellen Sie einen `params-only`-Validator zur Validierung von URL-Abfragezeichenfolgen-Parametern.
- Für die `POST`-Methode erstellen Sie einen `body-only`-Validator, um den Anforderungstext zu validieren.

Dadurch können nur bestimmte API-Aufrufe an die API weitergeleitet werden.

Um einen AWS CloudFormation Stapel zu erstellen

Laden Sie die Vorlage zur App-Erstellung für AWS CloudFormation herunter und entpacken Sie sie.

Um die folgenden Schritte durchzuführen, benötigen Sie die AWS Command Line Interface (AWS CLI) Version 2.

Bei langen Befehlen wird ein Escape-Zeichen (\) verwendet, um einen Befehl auf mehrere Zeilen aufzuteilen.

> **Note**
>
> In Windows werden einige Bash-CLI-Befehle, die Sie häufig verwenden (z. B. `zip`), von den integrierten Terminals des Betriebssystems nicht unterstützt. Um eine in Windows integrierte Version von Ubuntu und Bash zu erhalten, installieren Sie das Windows-Subsystem für Linux. Die CLI-Beispielbefehle in diesem Handbuch verwenden die Linux-Formatierung. Befehle, die Inline-JSON-Dokumente enthalten, müssen neu formatiert werden, wenn Sie die Windows-CLI verwenden.

1. Verwenden Sie den folgenden Befehl, um den AWS CloudFormation Stack zu erstellen.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
 --template-body file://request-validation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Verwenden Sie den folgenden Befehl, um den Status Ihres AWS CloudFormation Stacks zu überprüfen.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Wenn der Status Ihres AWS CloudFormation Stacks lautet`StackStatus: "CREATE_COMPLETE"`, verwenden Sie den folgenden Befehl, um relevante Ausgabewerte für future Schritte abzurufen.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

Die Ausgabewerte sind die folgenden:

- ApiId, das ist die ID für die API. Für dieses Tutorial lautet die API-ID `abc123`.
- ResourceId, das ist die ID für die Validator-Ressource, in der die `POST` Methoden `GET` und verfügbar gemacht werden. Für dieses Tutorial lautet die Ressourcen-ID `efg456`.

So erstellen Sie die Anforderungsvalidierungen und importieren ein Modell

1. Ein Validator ist erforderlich, um die Anforderungsvalidierung mit der AWS CLI zu verwenden. Verwenden Sie den folgenden Befehl, um einen Validator zu erstellen, der nur Anforderungsparameter validiert.

   ```
   aws apigateway create-request-validator --rest-api-id abc123 \
         --no-validate-request-body \
         --validate-request-parameters \
         --name params-only
   ```

   Notieren Sie sich die ID des `params-only`-Validators.

2. Verwenden Sie den folgenden Befehl, um einen Validator zu erstellen, der nur den Anforderungstext validiert.

   ```
   aws apigateway create-request-validator --rest-api-id abc123 \
         --validate-request-body \
         --no-validate-request-parameters \
         --name body-only
   ```

   Notieren Sie sich die ID des `body-only`-Validators.

3. Ein Modell ist erforderlich, um die Anforderungsvalidierung für den Text einer eingehenden Anfrage zu verwenden. Verwenden Sie den folgenden Befehl, um ein Modell zu importieren.

   ```
   aws apigateway create-model --rest-api-id abc123 --name PetStoreModel --description
    'My PetStore Model' --content-type 'application/json' --schema '{"type":
    "object", "required" : [ "name", "price", "type" ], "properties" : { "id" :
    {"type" : "integer"},"type" : {"type" : "string", "enum" : [ "dog", "cat",
    "fish" ]},"name" : { "type" : "string"},"price" : {"type" : "number","minimum" :
    25.0, "maximum" : 500.0}}}}'
   ```

Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

So erstellen Sie die Methoden **`GET`** und **`POST`**

1. Rufen Sie den folgenden Befehl auf, um die `GET`-HTTP-Methode der `/validate`-Ressource hinzuzufügen. Dieser Befehl erstellt die `GET`-Methode, fügt den `params-only`-Validator hinzu und legt die Abfragezeichenfolge `petType` nach Bedarf fest.

   ```
   aws apigateway put-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET \
          --authorization-type "NONE" \
          --request-validator-id aaa111 \
          --request-parameters "method.request.querystring.petType=true"
   ```

   Rufen Sie den folgenden Befehl auf, um die `POST`-HTTP-Methode der `/validate`-Ressource hinzuzufügen. Mit diesem Befehl wird die `POST`-Methode erstellt, der `body-only`-Validator hinzugefügt und das Modell an den Validator angehängt, der nur für den Text gilt.

   ```
   aws apigateway put-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method POST \
          --authorization-type "NONE" \
          --request-validator-id bbb222 \
          --request-models 'application/json'=PetStoreModel
   ```

2. Führen Sie den folgenden Befehl aus, um die `200 OK`-Antwort der `GET /validate`-Methode einzurichten.

   ```
   aws apigateway put-method-response --rest-api-id abc123  \
              --resource-id efg456 \
              --http-method GET \
              --status-code 200
   ```

   Führen Sie den folgenden Befehl aus, um die `200 OK`-Antwort der `POST /validate`-Methode einzurichten.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200
```

3. Führen Sie den folgenden Befehl aus, um einen `Integration` mit einem angegebenen HTTP-Endpunkt für die `GET /validation`-Methode einzurichten.

```
aws apigateway put-integration --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --type HTTP \
            --integration-http-method GET \
            --request-parameters '{"integration.request.querystring.type" :
 "method.request.querystring.petType"}' \
            --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

   Führen Sie den folgenden Befehl aus, um einen `Integration` mit einem angegebenen HTTP-Endpunkt für die `POST /validation`-Methode einzurichten.

```
aws apigateway put-integration --rest-api-id abc123  \
              --resource-id efg456 \
              --http-method POST \
              --type HTTP \
              --integration-http-method GET \
              --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

4. Führen Sie den folgenden Befehl aus, um die Integration der `GET /validation`-Methode einzurichten.

```
aws apigateway put-integration-response --rest-api-id abc123 \
              --resource-id efg456\
              --http-method GET \
              --status-code 200 \
              --selection-pattern ""
```

   Führen Sie den folgenden Befehl aus, um die Integration der `POST /validation`-Methode einzurichten.

```
aws apigateway put-integration-response --rest-api-id abc123 \
```

```
            --resource-id efg456 \
            --http-method POST \
            --status-code 200 \
            --selection-pattern ""
```

So testen Sie die API-Daten

1. Verwenden Sie den folgenden Befehl, um die GET-Methode zu testen, die eine Anforderungsvalidierung für die Abfragezeichenfolgen durchführt:

   ```
   aws apigateway test-invoke-method --rest-api-id abc123 \
               --resource-id efg456 \
               --http-method GET \
               --path-with-query-string '/validate?petType=dog'
   ```

   Das Ergebnis gibt `200 OK` und eine Liste von Hunden zurück.

2. Verwenden Sie den folgenden Befehl, um zu testen, ohne die Abfragezeichenfolge `petType` einzuschließen.

   ```
   aws apigateway test-invoke-method --rest-api-id abc123 \
               --resource-id efg456 \
               --http-method GET
   ```

   Das Ergebnis gibt einen `400`-Fehler zurück.

3. Verwenden Sie den folgenden Befehl, um die POST-Methode zu testen, die eine Anforderungsvalidierung für den Anforderungstext durchführt:

   ```
   aws apigateway test-invoke-method --rest-api-id abc123 \
               --resource-id efg456 \
               --http-method POST \
               --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
   ```

   Das Ergebnis gibt `200 OK` und eine Erfolgsmeldung zurück.

4. Verwenden Sie den folgenden Befehl, um mit einem ungültigen Text zu testen.

   ```
   aws apigateway test-invoke-method --rest-api-id abc123 \
                --resource-id efg456 \
                --http-method POST \
   ```

```
                --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

Das Ergebnis gibt einen `400`-Fehler zurück, da der Preis des Hundes über dem vom Modell festgelegten Höchstpreis liegt.

Um einen Stapel zu löschen AWS CloudFormation

- Verwenden Sie den folgenden Befehl, um Ihre AWS CloudFormation Ressourcen zu löschen.

```
aws cloudformation delete-stack  --stack-name request-validation-tutorial-cli
```

Einrichten der grundlegenden Anforderungsvalidierung mithilfe einer OpenAPI-Definition

Sie können eine Anforderungsvalidierung auf API-Ebene deklarieren, indem Sie eine Reihe von x-amazon-apigateway-request-Validators.RequestValidator-Objekt-Objekten in der x-amazon-apigateway-request-validators-Objekt-Zuordnung angeben, um auszuwählen, welcher Teil der Anfrage validiert werden soll. In der OpenAPI-Beispieldefinition gibt es zwei Validatoren:

- `all`-Validator, der sowohl den Text anhand des `RequestBodyModel`-Datenmodells als auch die Parameter validiert.
- `param-only`, der nur die Parameter validiert.

Zum Aktivieren einer Anforderungsvalidierung für alle Methoden einer API geben Sie eine x-amazon-apigateway-request-validator-Eigenschaft-Eigenschaft auf API-Ebene in der OpenAPI-Definitionsdatei an. In der Beispiel-OpenAPI-Definition wird der `all` für alle API-Methoden verwendet (sofern nicht anderweitig außer Kraft gesetzt). Wenn ein Modell zur Validierung des Textes verwendet wird und kein übereinstimmender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

Zum Aktivieren einer Anforderungsvalidierung für eine einzelne Methode geben Sie die `x-amazon-apigateway-request-validator`-Eigenschaft auf Methodenebene an. Im Beispiel mit OpenAPI-Definition überschreibt der `param-only`-Validator den `all`-Validator für die `GET`-Methode.

Um das OpenAPI-Beispiel in API Gateway zu importieren, lesen Sie die folgenden Anweisungen zu Importieren Sie eine regionale API in API Gateway oder zu Edge-optimierte API in API Gateway importieren.

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "/v1"
      }
    }
  } ],
  "paths" : {
    "/validation" : {
      "get" : {
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
```

```
              "schema" : {
                "$ref" : "#/components/schemas/ArrayOfError"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "params-only",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/xml" : "xml 400 response template",
              "application/json" : "json 400 response template"
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.querystring.type" : "method.request.querystring.q1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    },
    "post" : {
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "requestBody" : {
```

```
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/RequestBodyModel"
              }
            }
          },
          "required" : true
        },
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/ArrayOfError"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
```

```
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "RequestBodyModel" : {
        "required" : [ "name", "price", "type" ],
        "type" : "object",
        "properties" : {
          "id" : {
            "type" : "integer"
          },
          "type" : {
            "type" : "string",
            "enum" : [ "dog", "cat", "fish" ]
          },
          "name" : {
            "type" : "string"
          },
          "price" : {
            "maximum" : 500.0,
            "minimum" : 25.0,
            "type" : "number"
          }
        }
      },
      "ArrayOfError" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/components/schemas/Error"
        }
      },
      "Error" : {
        "type" : "object"
```

```
      }
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0.0",
    "title" : "ReqValidators Sample"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/validation" : {
      "get" : {
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
```

```
              "type" : "string"
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "params-only",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/xml" : "xml 400 response template",
              "application/json" : "json 400 response template"
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.querystring.type" : "method.request.querystring.q1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    },
    "post" : {
      "consumes" : [ "application/json" ],
      "produces" : [ "application/json", "application/xml" ],
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "type" : "string"
      }, {
        "in" : "body",
        "name" : "RequestBodyModel",
        "required" : true,
```

```
          "schema" : {
            "$ref" : "#/definitions/RequestBodyModel"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
```

```
    }
  },
  "definitions" : {
    "RequestBodyModel" : {
      "type" : "object",
      "required" : [ "name", "price", "type" ],
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string",
          "enum" : [ "dog", "cat", "fish" ]
        },
        "name" : {
          "type" : "string"
        },
        "price" : {
          "type" : "number",
          "minimum" : 25.0,
          "maximum" : 500.0
        }
      }
    },
    "ArrayOfError" : {
      "type" : "array",
      "items" : {
        "$ref" : "#/definitions/Error"
      }
    },
    "Error" : {
      "type" : "object"
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
```

```
}
```

OpenAPI-Definitionen einer Beispiel-API mit grundlegender Anforderungsvalidierung

Die folgende OpenAPI-Definition gilt für eine Beispiel-API mit aktivierter Anforderungsvalidierung. Die API ist eine Teilmenge der PetStore API. Mithilfe einer `POST`-Methode wird ein Haustier zur Sammlung `pets` hinzugefügt und mit einer `GET`-Methode erfolgt die Abfrage der Haustiere nach Typ.

In der Zuweisung `x-amazon-apigateway-request-validators` sind auf API-Ebene zwei Anforderungsvalidierungen deklariert. Die Validierung `params-only` ist für die API aktiviert und wird von der `GET`-Methode geerbt. Mit diesem Validator kann API Gateway überprüfen, ob der erforderliche Abfrageparameter (`q1`) in der eingehenden Anfrage enthalten und nicht leer ist. Die Validierung `all` ist für die `POST`-Methode aktiviert. Mit dieser Validierung wird geprüft, ob der erforderliche Header-Parameter (`h1`) gesetzt und nicht leer ist. Es wird auch überprüft, ob das Payload-Format dem angegebenen `RequestBodyModel` entspricht. Wenn kein passender Inhaltstyp gefunden wird, wird die Anforderungsüberprüfung nicht durchgeführt. Wenn ein Modell zur Validierung des Textes verwendet wird und kein übereinstimmender Inhaltstyp gefunden wird, wird die Anforderungsvalidierung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp zu verwenden, geben Sie es `$default` als Schlüssel an.

Bei diesem Modell muss das JSON-Objekt der Eingabe die Eigenschaften `name`, `type` und `price` enthalten. Die Eigenschaft `name` kann eine beliebige Zeichenfolge sein, `type` muss einem der angegebenen Aufzählungsfelder (`["dog", "cat", "fish"]`) entsprechen und `price` muss zwischen 25 und 500 liegen. Der Parameter `id` ist nicht erforderlich.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "title": "ReqValidators Sample",
    "version": "1.0.0"
  },
  "schemes": [
    "https"
  ],
  "basePath": "/v1",
  "produces": [
    "application/json"
  ],
```

```
"x-amazon-apigateway-request-validators" : {
  "all" : {
    "validateRequestBody" : true,
    "validateRequestParameters" : true
  },
  "params-only" : {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
  }
},
"x-amazon-apigateway-request-validator" : "params-only",
"paths": {
  "/validation": {
    "post": {
      "x-amazon-apigateway-request-validator" : "all",
      "parameters": [
        {
          "in": "header",
          "name": "h1",
          "required": true
        },
        {
          "in": "body",
          "name": "RequestBodyModel",
          "required": true,
          "schema": {
            "$ref": "#/definitions/RequestBodyModel"
          }
        }
      ],
      "responses": {
        "200": {
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Error"
            }
          },
          "headers" : {
            "test-method-response-header" : {
              "type" : "string"
            }
          }
        }
```

```
        },
        "security" : [{
          "api_key" : []
        }],
        "x-amazon-apigateway-auth" : {
          "type" : "none"
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "httpMethod" : "POST",
          "requestParameters": {
            "integration.request.header.custom_h1": "method.request.header.h1"
          },
          "responses" : {
            "2\\d{2}" : {
              "statusCode" : "200"
            },
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/json" : "json 400 response template",
                "application/xml" : "xml 400 response template"
              }
            }
          }
        }
      },
      "get": {
        "parameters": [
          {
            "name": "q1",
            "in": "query",
            "required": true
          }
        ],
        "responses": {
          "200": {
            "schema": {
              "type": "array",
```

```
            "items": {
              "$ref": "#/definitions/Error"
            }
          },
          "headers" : {
            "test-method-response-header" : {
              "type" : "string"
            }
          }
        }
      },
      "security" : [{
        "api_key" : []
      }],
      "x-amazon-apigateway-auth" : {
        "type" : "none"
      },
      "x-amazon-apigateway-integration" : {
        "type" : "http",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "httpMethod" : "GET",
        "requestParameters": {
          "integration.request.querystring.type": "method.request.querystring.q1"
        },
        "responses" : {
          "2\\d{2}" : {
            "statusCode" : "200"
          },
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/json" : "json 400 response template",
              "application/xml" : "xml 400 response template"
            }
          }
        }
      }
    }
  },
```

```
  "definitions": {
    "RequestBodyModel": {
      "type": "object",
      "properties": {
        "id":    { "type": "integer" },
        "type":  { "type": "string", "enum": ["dog", "cat", "fish"] },
        "name":  { "type": "string" },
        "price": { "type": "number", "minimum": 25, "maximum": 500 }
      },
      "required": ["type", "name", "price"]
    },
    "Error": {
      "type": "object",
      "properties": {

      }
    }
  }
}
```

AWS CloudFormation Vorlage einer Beispiel-API mit grundlegender Anforderungsvalidierung

Die folgende AWS CloudFormation Beispielvorlagendefinition definiert eine Beispiel-API mit aktivierter Anforderungsvalidierung. Die API ist eine Teilmenge der PetStoreAPI. Mithilfe einer `POST`-Methode wird ein Haustier zur Sammlung `pets` hinzugefügt und mit einer `GET`-Methode erfolgt die Abfrage der Haustiere nach Typ.

Es wurden zwei Anforderungsvalidatoren deklariert:

`GETValidator`

Dieser Validator ist für die `GET`-Methode aktiviert. Damit kann API Gateway überprüfen, ob der erforderliche Abfrageparameter (`q1`) in der eingehenden Anfrage enthalten und nicht leer ist.

`POSTValidator`

Dieser Validator ist für die `POST`-Methode aktiviert. Damit kann API Gateway prüfen, ob das Nutzlast-Anforderungsformat dem angegebenen `RequestBodyModel` entspricht. Wenn der Inhaltstyp `application/json` ist und kein passender Inhaltstyp gefunden wird, wird die Anforderungsüberprüfung nicht durchgeführt. Um das gleiche Modell unabhängig vom Inhaltstyp

zu verwenden, geben Sie `$default` an. `RequestBodyModel` enthält ein zusätzliches Modell, `RequestBodyModelId`, um die Pet-ID zu definieren.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: ReqValidatorsSample
  RequestBodyModelId:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModelId
        properties:
            id:
              type: integer
  RequestBodyModel:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet type, name, price, and ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModel
        required:
          - price
          - name
          - type
        type: object
        properties:
            id:
```

```
              "$ref": !Sub
                - 'https://apigateway.amazonaws.com/restapis/${Api}/models/
${RequestBodyModelId}'
                - Api: !Ref Api
                  RequestBodyModelId: !Ref RequestBodyModelId
            price:
              type: number
              minimum: 25
              maximum: 500
            name:
              type: string
            type:
              type: string
              enum:
                - "dog"
                - "cat"
                - "fish"
  GETValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: params-only
      RestApiId: !Ref Api
      ValidateRequestBody: False
      ValidateRequestParameters: True
  POSTValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: body-only
      RestApiId: !Ref Api
      ValidateRequestBody: True
      ValidateRequestParameters: False
  ValidationResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'validation'
  ValidationMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: GET
      AuthorizationType: NONE
```

```
      RequestValidatorId: !Ref GETValidator
      RequestParameters:
        method.request.querystring.q1: true
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ValidationMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: POST
      AuthorizationType: NONE
      RequestValidatorId: !Ref POSTValidator
      RequestModels:
        application/json : !Ref RequestBodyModel
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: POST
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - ValidationMethodGet
      - RequestBodyModel
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Datentransformationen für REST-APIs einrichten

In API Gateway kann eine API-Methodenanforderung eine Nutzlast in einem anderen Format als dem der entsprechenden Integrationsanforderungsnutzlast entgegennehmen. Das Backend wiederum kann eine Integrationsantwortnutzlast zurückgeben, die nicht der Methodenantwortnutzlast entspricht. Mithilfe von Zuweisungsvorlagen können Sie URL-Pfadparameter, URL-Abfragezeichenfolgen-Parameter, HTTP-Header und den Anforderungstext im gesamten API Gateway zuordnen.

Eine Mapping-Vorlage ist ein in Velocity Template Language (VTL) ausgedrücktes Skript, dass mithilfe von JSONPath-Ausdrücken auf die Nutzlast angewendet wird.

Die Nutzlast kann ein dem JSON-Schema Entwurf 4 entsprechendes Datenmodell umfassen. Weitere Informationen zu Modellen finden Sie unter Datenmodelle.

> **Note**
>
> Sie müssen kein Modell definieren, um eine Zuweisungsvorlage zu erstellen, aber Sie müssen ein Modell definieren, damit API Gateway ein SDK generiert oder die Anforderungstextvalidierung für Ihre API aktiviert.

Themen

Grundlegendes zu Zuweisungsvorlagen

In API Gateway kann die Methodenanforderung oder -antwort einer API eine Nutzlast in einem anderen Format als die Integrationsanfrage oder -antwort annehmen.

Sie können Ihre Daten folgendermaßen transformieren:

- Anpassung der Nutzlast an ein API-spezifiziertes Format
- Überschreiben der Anforderungs- und Antwortparameter und Statuscodes einer API-Anfrage
- Rückgabe der vom Client ausgewählten Antwort-Header
- Ordnen Sie Pfadparameter, Abfragezeichenfolge-Parameter oder Header-Parameter in der Methodenanforderung des HTTP-Proxys oder AWS-Service Proxys zu.

- Wählen Sie aus, welche Daten mithilfe der Integration gesendet werden sollen AWS-Services, z. B. mit Amazon DynamoDB- oder Lambda-Funktionen oder HTTP-Endpunkten.

Sie können Zuweisungsvorlagen verwenden, um Ihre Daten zu transformieren. Eine Zuweisungsvorlage ist ein in Velocity Template Language (VTL) ausgedrücktes Skript, das mithilfe von JSONPath-Ausdrücken auf die Nutzlast angewendet wird.

Das folgende Beispiel zeigt Eingabedaten, eine Zuordnungsvorlage und Ausgabedaten für eine Transformation der Daten. PetStore

Eingabedaten

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

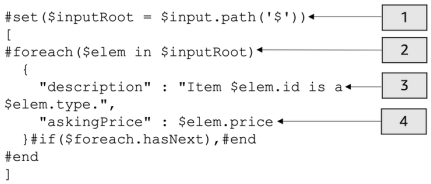
Zuweisungsvorlage

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end

#end
]
```

Ausgaben

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Das folgende Diagramm zeigt Details dieser Zuweisungsvorlage.



1. Die Variable `$inputRoot` stellt in den ursprünglichen JSON-Daten aus dem vorherigen Abschnitt das Stammobjekt dar. Richtlinien beginnen mit dem #-Symbol.
2. Eine `foreach`-Schleife durchläuft jedes Objekt in den ursprünglichen JSON-Daten.
3. Die Beschreibung ist eine Verkettung der Pet-`id` und des `type` der ursprünglichen JSON-Daten.
4. `askingPrice` ist der `price` aus den ursprünglichen JSON-Daten.

PetStore Vorlage für eine Zuordnung

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end
#end
```

```
9 ]
```

In dieser Zuweisungsvorlage gilt:

1. Die Variable `$inputRoot` in Zeile 1 stellt das Stammobjekt in den ursprünglichen JSON-Daten aus dem vorherigen Abschnitt dar. Richtlinien beginnen mit dem #-Symbol.
2. In Zeile 3 durchläuft eine `foreach`-Schleife jedes Objekt in den ursprünglichen JSON-Daten.
3. In Zeile 5 ist die `description` eine Verkettung der Pet-`id` und der ursprünglichen `type`-JSON-Daten.
4. In Zeile 6 ist `askingPrice` der `price` aus den ursprünglichen JSON-Daten.

Weitere Informationen zur Velocity Template Language finden Sie in der Apache Velocity – VTL-Reference. Weitere Informationen zu JSONPath finden Sie unter JSONPath – XPath for JSON.

Die Mapping-Vorlage geht davon aus, dass die zugrundeliegenden Daten zu einem JSON-Objekt gehören. Es ist nicht erforderlich, ein Modell für die Daten zu definieren. Ein Modell für die Ausgabedaten ermöglicht es jedoch, vorherige Daten als sprachspezifisches Objekt zurückzugeben. Weitere Informationen finden Sie unter Datenmodelle.

Komplexe Zuweisungsvorlagen

Sie können auch kompliziertere Zuordnungsvorlagen erstellen. Das folgende Beispiel zeigt die Verkettung von Referenzen und einen Grenzwert von 100, um festzustellen, ob Sie sich ein Haustier leisten können.

Eingabedaten

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

```
    "price": 0.99
  }
]
```

Zuweisu
svorlage

```
#set($inputRoot = $input.path('$'))
#set($cheap = 100)
[
#foreach($elem in $inputRoot)
  {
#set($name = "${elem.type}number$elem.id")
    "name" : $name,
    "description" : "Item $elem.id is a $elem.type.",
    #if($elem.price > $cheap )#set ($afford = 'too much!') #{else}#set
 ($afford = $elem.price)#end
"askingPrice" : $afford
  }#if($foreach.hasNext),#end

#end
]
```

Ausgab
ten

```
[
  {
    "name" : dognumber1,
    "description" : "Item 1 is a dog.",
    "askingPrice" : too much!
  },
  {
    "name" : catnumber2,
    "description" : "Item 2 is a cat.",
    "askingPrice" : too much!
  },
  {
    "name" : fishnumber3,
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Sie können sich auch kompliziertere Datenmodelle ansehen. Siehe Beispieldatenmodelle und Zuordnungsvorlagen für API Gateway.

Einrichten von Datentransformationen in API Gateway

In diesem Abschnitt wird gezeigt, wie Sie Mapping-Vorlagen einrichten, um Integrationsanfragen und -antworten mithilfe der Konsole und der AWS CLI zu transformieren.

Themen

Einrichten einer Datentransformation mithilfe der API-Gateway-Konsole

In diesem Tutorial erstellen Sie eine unvollständige API- und DynamoDB-Tabelle mithilfe der folgenden ZIP-Datei .zip. data-transformation-tutorial-console Diese unvollständige API hat eine `/pets`-Ressource mit den Methoden `GET` und `POST`.

- Die `GET`-Methode ruft Daten vom `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt ab. Die Ausgabedaten werden gemäß der Zuweisungsvorlage zu PetStore Vorlage für eine Zuordnung transformiert.
- Die `POST`-Methode ermöglicht dem Benutzer die `POST`-Aktion von Haustierinformationen mithilfe einer Zuweisungsvorlage an eine Amazon-DynamoDB-Tabelle.

Laden Sie die Vorlage zur App-Erstellung für herunter und entpacken Sie sie. AWS CloudFormation Sie verwenden diese Vorlage, um eine DynamoDB-Tabelle zum Posten von Haustierinformationen und einer unvollständigen API zu erstellen. Sie schließen die restlichen Schritte in der API-Gateway-Konsole ab.

Um einen Stack zu erstellen AWS CloudFormation

1. Öffnen Sie die AWS CloudFormation Konsole unter https://console.aws.amazon.com/cloudformation.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard) aus.
3. Wählen Sie unter Vorlage angeben die Option Vorlagendatei hochladen aus.
4. Wählen Sie die Vorlage aus, die Sie heruntergeladen haben.
5. Wählen Sie Weiter aus.

6. Geben Sie für Stack-Name die Zeichenfolge **`data-transformation-tutorial-console`** ein und klicken Sie auf Weiter.
7. Wählen Sie in Stack-Optionen konfigurieren die Option Weiter aus.
8. Bestätigen Sie bei Capabilities, dass IAM-Ressourcen in Ihrem Konto erstellt werden AWS CloudFormation können.
9. Wählen Sie Absenden aus.

AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Wenn der Status Ihres AWS CloudFormation Stacks CREATE_COMPLETE lautet, können Sie mit dem nächsten Schritt fortfahren.

So testen Sie die **`GET`**-Integrationsantwort

1. Wählen Sie auf der Registerkarte Ressourcen des AWS CloudFormation Stacks für **`data-transformation-tutorial-console`** die physische ID Ihrer API aus.
2. Klicken Sie im Hauptnavigationsbereich auf Ressourcen und wählen Sie dann die GET-Methode in der Ressourcenstruktur aus.
3. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

   Das Ergebnis des Tests wird Folgendes zeigen:

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

```
]
```

Sie transformieren diese Ausgabe gemäß der Zuweisungsvorlage in PetStore Vorlage für eine Zuordnung.

So transformieren Sie die **GET**-Integrationsantwort

1. Klicken Sie auf die Registerkarte Integrationsantwort.

   Derzeit sind keine Zuweisungsvorlagen definiert, sodass die Integrationsantwort nicht transformiert wird.
2. Klicken Sie unter Standard - Antwort auf Bearbeiten.
3. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:

   a. Wählen Sie Add mapping template.

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

   c. Geben Sie für Vorlagentext Folgendes ein:

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end

#end
]
```

   Wählen Sie Speichern.

So testen Sie die **GET**-Integrationsantwort

- Wählen Sie die Registerkarte Test und dann Test aus

  Die Ausgabe des Tests zeigt die transformierte Antwort.

```
[
```

```
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

So transformieren Sie Eingabedaten aus der **POST**-Methode

1. Wählen Sie die POST-Methode aus.
2. Wählen Sie die Registerkarte Integrationsanfrage aus und klicken Sie dann unter Einstellungen für Integrationsanforderungen auf Bearbeiten.

   Die AWS CloudFormation Vorlage hat einige Felder für Integrationsanfragen ausgefüllt.

   - Der Integrationstyp ist AWS-Service.
   - Das AWS-Service ist DynamoDB.
   - Die HTTP-Methode ist `POST`.
   - Die Aktion ist `PutItem`.
   - Die Ausführungsrolle, die es API Gateway ermöglicht, ein Element in die DynamoDB-Tabelle aufzunehmen, lautet. `data-transformation-tutorial-console-APIGatewayRole` AWS CloudFormation hat diese Rolle erstellt, damit API Gateway über die Mindestberechtigungen für die Interaktion mit DynamoDB verfügt.

   Der Name der DynamoDB-Tabelle wurde nicht angegeben. In den folgenden Schritten geben Sie den Namen an.
3. Für Text-Passthrough anfordern wählen Sie Nie aus.

   Dies bedeutet, dass die API Daten mit Inhaltstypen ablehnt, die keine Zuweisungsvorlage haben.
4. Wählen Sie Zuordnungsvorlagen aus.

5. Der Inhaltstyp ist auf `application/json` gesetzt. Das bedeutet, dass alle Inhaltstypen, die nicht „application/json“ sind, von der API abgelehnt werden. Weitere Informationen zu Integrations-Passthrough-Verhaltensweisen finden Sie unter Integrations-Pass-Through-Verhalten.
6. Geben Sie den folgenden Code in den Texteditor ein.

```
{
    "TableName":"data-transformation-tutorial-console-ddb",
    "Item": {
        "id": {
            "N": $input.json("$.id")
        },
        "type": {
            "S": $input.json("$.type")
        },
        "price": {
            "N": $input.json("$.price")
        }
    }
}
```

   Diese Vorlage spezifiziert die Tabelle als `data-transformation-tutorial-console-ddb` und legt die Elemente als `id`, `type` und `price` fest. Die Elemente kommen aus dem Hauptteil der POST-Methode. Sie können auch ein Datenmodell verwenden, um eine Zuweisungsvorlage zu erstellen. Weitere Informationen finden Sie unter Verwenden der Anforderungsvalidierung in API Gateway.
7. Betätigen Sie die Schaltfläche Speichern, um die Zuweisungsvorlage zu speichern.

So fügen Sie eine Methode und eine Integrationsantwort aus der **POST**-Methode hinzu

Sie AWS CloudFormation hat eine leere Methode und eine leere Integrationsantwort erstellt. Sie werden diese Antwort bearbeiten, um weitere Informationen bereitzustellen. Weitere Informationen zum Bearbeiten von Antworten finden Sie unter Daten-Mapping-Referenz für Amazon API Gateway-API-Anfragen und Antworten.

1. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
2. Wählen Sie Zuordnungsvorlagen aus und klicken Sie dann auf Zuordnungsvorlage hinzufügen.
3. Geben Sie für Inhaltstyp **`application/json`** ein.

4. Geben Sie im Code-Editor die folgende Ausgabe–Zuweisungsvorlage ein, um eine Ausgabenachricht zu senden:

```
{ "message" : "Your response was recorded at $context.requestTime" }
```

   Weitere Informationen zu Kontextvariablen finden Sie unter `$context`Variablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch .

5. Betätigen Sie die Schaltfläche Speichern, um die Zuweisungsvorlage zu speichern.

Testen der **POST**-Methode

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

1. Geben Sie im Anforderungstext das folgende Beispiel ein.

```
{
          "id": "4",
          "type" : "dog",
          "price": "321"
}
```

2. Wählen Sie Test aus.

   Die Ausgabe sollte Ihre Erfolgsmeldung enthalten.

   Sie können die DynamoDB-Konsole unter https://console.aws.amazon.com/dynamodb/ öffnen, um zu überprüfen, ob sich das Beispielelement in Ihrer Tabelle befindet.

Um einen AWS CloudFormation Stapel zu löschen

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

Datentransformation mit der AWS CLI einrichten

In diesem Tutorial erstellen Sie eine unvollständige API- und DynamoDB-Tabelle mithilfe der folgenden ZIP-Datei .zip. data-transformation-tutorial-cli Diese unvollständige API verfügt über eine `/pets`-Ressource mit einer `GET`-Methode, die in den `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt integriert ist. Sie erstellen eine `POST`-Methode, um eine Verbindung zu einer DynamoDB-Tabelle herzustellen, und verwenden Zuweisungsvorlagen, um Daten in eine DynamoDB-Tabelle einzugeben.

- Sie transformieren die Ausgabedaten gemäß der Zuweisungsvorlage zu PetStore Vorlage für eine Zuordnung.
- Sie erstellen eine `POST`-Methode, die es dem Benutzer ermöglicht, mithilfe einer Zuweisungsvorlage die `POST`-Aktion an eine Amazon-DynamoDB-Tabelle durchzuführen.

Um einen Stapel zu erstellen AWS CloudFormation

Laden Sie die Vorlage zur App-Erstellung für AWS CloudFormation herunter und entpacken Sie sie.

Um die folgenden Schritte durchzuführen, benötigen Sie die AWS Command Line Interface (AWS CLI) Version 2.

Bei langen Befehlen wird ein Escape-Zeichen (\) verwendet, um einen Befehl auf mehrere Zeilen aufzuteilen.

> **Note**
>
> In Windows werden einige Bash-CLI-Befehle, die Sie häufig verwenden (z. B. `zip`), von den integrierten Terminals des Betriebssystems nicht unterstützt. Um eine in Windows integrierte Version von Ubuntu und Bash zu erhalten, installieren Sie das Windows-Subsystem für Linux. Die CLI-Beispielbefehle in diesem Handbuch verwenden die Linux-Formatierung. Befehle, die Inline-JSON-Dokumente enthalten, müssen neu formatiert werden, wenn Sie die Windows-CLI verwenden.

1. Verwenden Sie den folgenden Befehl, um den AWS CloudFormation Stack zu erstellen.

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
 --template-body file://data-transformation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation stellt die in der Vorlage angegebenen Ressourcen bereit. Die Bereitstellung der Ressourcen kann einige Minuten dauern. Verwenden Sie den folgenden Befehl, um den Status Ihres AWS CloudFormation Stacks zu überprüfen.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Wenn der Status Ihres AWS CloudFormation Stacks lautet`StackStatus: "CREATE_COMPLETE"`, verwenden Sie den folgenden Befehl, um relevante Ausgabewerte für future Schritte abzurufen.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   Die Ausgabewerte sind die folgenden:

   - ApiRole, das ist der Rollenname, der es API Gateway ermöglicht, Elemente in die DynamoDB-Tabelle einzufügen. Für dieses Tutorial ist der Rollenname `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
   - DDBTableName, das ist der Name der DynamoDB-Tabelle. Für dieses Tutorial ist der Name der Tabelle `data-transformation-tutorial-cli-ddb`.
   - ResourceId, das ist die ID der Pets-Ressource, in der die `POST` Methoden `GET` und verfügbar sind. Für dieses Tutorial lautet die Ressourcen-ID `efg456`.
   - ApiId, was die ID für die API ist. Für dieses Tutorial lautet die API-ID `abc123`.

So testen Sie die **`GET`**-Methode vor der Datentransformation

- Geben Sie den folgenden Befehl ein, um die `GET`-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET
```

   Die Ausgabe des Tests wird Folgendes zeigen.

```
[
  {
    "id": 1,
```

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

Sie transformieren diese Ausgabe gemäß der Zuweisungsvorlage in PetStore Vorlage für eine Zuordnung.

So transformieren Sie die **GET**-Integrationsantwort

- Führen Sie den folgenden Befehl aus, um die Integrationsantwort für die GET-Methode zu aktualisieren. Ersetzen Sie die *rest-api-id*und *resource-id* durch Ihre Werte.

  Verwenden Sie den folgenden Befehl, um die Integrationsantwort zu erstellen.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"}'
```

So testen Sie die **GET**-Methode

- Geben Sie den folgenden Befehl ein, um die GET-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
```

```
  --resource-id efg456 \
  --http-method GET \
```

Die Ausgabe des Tests zeigt die transformierte Antwort.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

So erstellen Sie eine **POST**-Methode

1. Führen Sie den folgenden Befehl aus, um eine neue Methode für die `/pets`-Ressource zu erstellen.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Mit dieser Methode können Sie Pet-Informationen an die DynamoDB-Tabelle senden, die Sie im Stack erstellt haben. AWS CloudFormation

2. Verwenden Sie den folgenden Befehl, um eine AWS-Service Integration für die `POST` Methode zu erstellen.

```
aws apigateway put-integration --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --type AWS \
  --integration-http-method POST \
```

```
  --uri "arn:aws:apigateway:us-east-2:dynamodb:action/PutItem" \
  --credentials arn:aws:iam::111122223333:role/data-transformation-tutorial-cli-
APIGatewayRole-ABCDEFG \
  --request-templates '{"application/json":"{\"TableName\":\"data-transformation-
tutorial-cli-ddb\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":
$input.json(\"$.type\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"}'
```

3. Verwenden Sie den folgenden Befehl, um eine Methodenantwort für einen erfolgreichen Aufruf der POST-Methode zu erstellen.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

4. Verwenden Sie den folgenden Befehl, um eine Integrationsantwort für den erfolgreichen Aufruf der POST-Methode zu erstellen.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "{\"message\": \"Your response was
 recorded at $context.requestTime\"}"}'
```

So testen Sie die **POST**-Methode

- Geben Sie den folgenden Befehl ein, um die POST-Methode zu testen.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{\"id\": \"4\", \"type\": \"dog\", \"price\": \"321\"}'
```

  In der Ausgabe wird die Erfolgsmeldung angezeigt.

Um einen AWS CloudFormation Stapel zu löschen

- Verwenden Sie den folgenden Befehl, um Ihre AWS CloudFormation Ressourcen zu löschen.

```
aws cloudformation delete-stack  --stack-name data-transformation-tutorial-cli
```

AWS CloudFormation Vorlage für die Datentransformation abgeschlossen

Das folgende Beispiel ist eine fertige AWS CloudFormation Vorlage, die eine API und eine DynamoDB-Tabelle mit einer `/pets` Ressource mit `GET` und `POST` Methoden erstellt.

- Die `GET`-Methode ruft Daten vom `http://petstore-demo-endpoint.execute-api.com/petstore/pets`-HTTP-Endpunkt ab. Die Ausgabedaten werden gemäß der Zuweisungsvorlage zu PetStore Vorlage für eine Zuordnung transformiert.
- Die `POST`-Methode ermöglicht es dem Benutzer, mithilfe einer Zuweisungsvorlage die `POST`-Aktion zu einer DynamoDB-Tabelle durchzuführen.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A completed Amazon API Gateway REST API that uses non-proxy integration
 to POST to an Amazon DynamoDB table and non-proxy integration to GET transformed pets
 data.
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  DynamoDBTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      TableName: !Sub data-transformation-tutorial-complete
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: N
      KeySchema:
        - AttributeName: id
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
  APIGatewayRole:
    Type: 'AWS::IAM::Role'
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - apigateway.amazonaws.com
      Policies:
        - PolicyName: APIGatewayDynamoDBPolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'dynamodb:PutItem'
                Resource: !GetAtt DynamoDBTable.Arn
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: data-transformation-complete-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        PassthroughBehavior: WHEN_NO_TEMPLATES
        IntegrationResponses:
```

```
          - StatusCode: '200'
            ResponseTemplates:
              application/json: "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"
      MethodResponses:
        - StatusCode: '200'
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: AWS
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: POST
        Uri: arn:aws:apigateway:us-west-1:dynamodb:action/PutItem
        PassthroughBehavior: NEVER
        RequestTemplates:
          application/json: "{\"TableName\":\"data-transformation-tutorial-complete
\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":$input.json(\"$.type
\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"
        IntegrationResponses:
          - StatusCode: 200
            ResponseTemplates:
              application/json: "{\"message\": \"Your response was recorded at
 $context.requestTime\"}"
      MethodResponses:
        - StatusCode: '200'

  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
Outputs:
  ApiId:
    Description: API ID for CLI commands
```

```
    Value: !Ref Api
  ResourceId:
    Description: /pets resource ID for CLI commands
    Value: !Ref PetsResource
  ApiRole:
    Description: Role ID to allow API Gateway to put and scan items in DynamoDB table
    Value: !Ref APIGatewayRole
  DDBTableName:
    Description: DynamoDB table name
    Value: !Ref DynamoDBTable
```

Nächste Schritte

Informationen zu komplexeren Mapping-Vorlagen finden Sie in den folgenden Beispielen:

- Komplexere Modelle und Zuweisungsvorlagen finden Sie im Beispiel-Fotoalbum Beispiel für ein Fotoalbum.
- Weitere Hinweise zu -Modellen finden Sie unter Datenmodelle.
- Informationen zur Zuordnung verschiedener Antwortcode-Ausgaben finden Sie unter Mappings von Anfrage- und Antwortdaten über die API Gateway-Konsole einrichten.
- Informationen zum Festlegen von Datenzuordnungen anhand der Methodenanforderungsdaten einer API finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen.

Verwendung einer Zuweisungsvorlage zum Überschreiben der Anforderungs- und Antwortparameter und Statuscodes einer API

Mit den standardmäßigen API Gateway Gateway-Vorlagen für die Zuordnung von Parametern und Antwortcodes können Sie Parameter zuordnen one-to-one und eine Familie von Integrationsantwortstatuscodes (denen ein regulärer Ausdruck entspricht) einem einzelnen Antwortstatuscode zuordnen. Überschreibungen von Zuordnungsvorlagen bieten Ihnen die Flexibilität, many-to-one Parameterzuordnungen durchzuführen, Parameter zu überschreiben, nachdem standardmäßige API-Gateway-Zuordnungen angewendet wurden, Parameter auf der Grundlage von Hauptinhalten oder anderen Parameterwerten bedingt zuzuordnen, neue Parameter im laufenden Betrieb programmgesteuert zu erstellen und Statuscodes zu überschreiben, die von Ihrem Integrationsendpunkt zurückgegeben werden. Alle Typen von Anforderungsparametern, Antwort-Headern oder Antwort-Statuscodes können überschrieben werden.

Im Folgenden finden Sie Beispielfälle für für eine Mapping-Vorlagen-Überschreibung:

- Um einen neuen Header als Kombination zweier Parameter zu erstellen (oder einen bestehenden Header zu überschreiben)
- Um den Antwortcode auf Basis des Body-Inhalts als Erfolgs- oder Fehlschlag-Code zu überschreiben
- Um einen Parameter basierend auf dessen Inhalten oder den Inhalten eines anderen Parameters bedingt neu zuzuweisen
- Um den Inhalt eines json-Body zu durchlaufen und Schlüsselwertpaare Headern oder Abfragestrings neu zuzuweisen

Um eine Mapping-Vorlagen-Überschreibung zu erstellen, verwenden Sie eine oder mehrere der folgenden `$context` Variablen in einer Mapping-Vorlage:

Mapping-Vorlage für Anforderungstext	Mapping-Vorlage für Antworttext
`$context.requestOverride.header.` *`header_name`*	`$context.responseOverride.header.` *`header_name`*
`$context.requestOverride.path.` *`path_name`*	`$context.responseOverride.status`
`$context.requestOverride.querystring.` *`querystring_name`*	

> **Note**
>
> Mapping-Vorlagen-Überschreibungen können nicht zusammen mit Proxy-Integrationsendpunkten verwendet werden, denen Daten-Mappings fehlen. Weitere Informationen zu den Integrationstypen finden Sie unter Auswählen eines API Gateway-API-Integration.

> **Important**
>
> Überschreibungen sind endgültig. Eine Überschreibung darf nur einmal auf jeden Parameter angewendet werden. Der Versuch, denselben Parameter mehrmals zu überschreiben, führt zu 5XX-Antworten von Amazon API Gateway. Wenn Sie denselben Parameter in einer

Vorlage mehrmals überschreiben müssen, empfehlen wir die Erstellung einer Variable und Umsetzung der Überschreibung am Ende der Vorlage. Beachten Sie, dass die Vorlage erst nach Einlesen der gesamten Vorlage angewendet wird. Siehe Tutorial: Anfrageparameter und -Header einer API mit der API Gateway-Konsole überschreiben.

Die folgenden Tutorials zeigen, wie man eine Mapping-Vorlagen-Überschreibung in der API Gateway-Konsole erstellt und testet. In diesen Tutorials wird die Beispiel-API als Ausgangspunkt verwendet. PetStore In beiden Tutorials wird davon ausgegangen, dass Sie die PetStore Beispiel-API bereits erstellt haben.

Themen

- Tutorial: Antwortstatuscodes einer API mit der API Gateway-Konsole überschreiben
- Tutorial: Anfrageparameter und -Header einer API mit der API Gateway-Konsole überschreiben
- Beispiele: Anfrageparameter und -Header einer API mit der API Gateway-CLI überschreiben
- Beispiel: Überschreiben Sie die Anforderungsparameter und Header einer API mithilfe des SDK für JavaScript

Tutorial: Antwortstatuscodes einer API mit der API Gateway-Konsole überschreiben

Um ein Haustier mithilfe der PetStore Beispiel-API abzurufen, verwenden Sie die API-Methode request of. `GET /pets/{petId}` Dabei `{petId}` handelt es sich um einen Pfadparameter, der zur Laufzeit eine Zahl annehmen kann.

In diesem Tutorial überschreiben Sie den Antwortcode dieser `GET`-Methode, indem Sie eine Mapping-Vorlage erstellen, die `$context.responseOverride.status 400` zuweist, wenn eine Fehlerbedingung auftritt.

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie unter APIs die PetStore API und dann Ressourcen aus.
3. Wählen Sie in der Ressourcen-Struktur unter `GET` die `/{petId}`-Methode aus.
4. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
5. Geben Sie für petId **`-1`** ein und klicken Sie dann auf Test.

In den Ergebnissen fällt Ihnen zweierlei auf:

Zunächst weist der Antworttext auf einen out-of-range Fehler hin:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Zweitens: Die letzte Zeile unter dem Feld Protokolle endet mit: `Method completed with status: 200`.

6. Klicken Sie auf der Registerkarte Integrationsantwort unter Standard - Antwort auf Bearbeiten.
7. Wählen Sie Zuordnungsvorlagen aus.
8. Wählen Sie Add mapping template.
9. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
10. Geben Sie für Vorlagentext Folgendes ein:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

11. Wählen Sie Speichern.
12. Wählen Sie die Registerkarte Test.
13. Geben Sie als petid **`-1`** ein.
14. In den Ergebnissen weist der Antworttext auf einen out-of-range Fehler hin:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
```

```
  ]
}
```

Die letzte Zeile unter dem Feld Logs endet nun jedoch mit: `Method completed with status: 400`.

Tutorial: Anfrageparameter und -Header einer API mit der API Gateway-Konsole überschreiben

In diesem Tutorial überschreiben Sie den Anfrageheadercode der `GET`-Methode, indem Sie eine Mapping-Vorlage erstellen, die `$context.requestOverride.header.`*`header_name`* einem neuen Header zuweist, der zwei andere Header kombiniert.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie unter APIs die PetStore API aus.
3. Wählen Sie in der Ressourcen-Struktur unter `GET` die `/pet`-Methode aus.
4. Klicken Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen auf Bearbeiten.
5. Wählen Sie HTTP Request Headers (HTTP-Anforderungs-Header) und dann Add header (Header hinzufügen) aus.
6. Geben Sie unter Name **`header1`** ein.
7. Klicken Sie auf Header hinzufügen und erstellen Sie dann einen zweiten Header namens **`header2`**.
8. Wählen Sie Speichern.
9. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
10. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.
11. Wählen Sie Vorlagen zuordnen aus und gehen Sie dann wie folgt vor:

    a. Wählen Sie Add mapping template.

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

    c. Geben Sie für Vorlagentext Folgendes ein:

```
#set($header1Override = "foo")
```

```
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.requestOverride.header.header3 = $header3Value)
#set($context.requestOverride.header.header1 = $header1Override)
#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])
```

12. Wählen Sie Speichern.
13. Wählen Sie die Registerkarte Test.
14. Kopieren Sie unter Headers für {pets} den folgenden Code:

```
header1:header1Val
header2:header2Val
```

15. Wählen Sie Test aus.

    Im Protokoll sollten Sie jetzt einen Eintrag mit diesem Text sehen:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=foo, x-amzn-apigateway-api-id=<api-id>,
Accept=application/json, multivalueheader=foo,header1Valheader2Val}
```

Beispiele: Anfrageparameter und -Header einer API mit der API Gateway-CLI überschreiben

Das folgende CLI-Beispiel demonstriert, wie Sie mit dem `put-integration`-Befehl einen Antwortcode überschreiben:

```
aws apigateway put-integration --rest-api-id <API_ID> --resource-
id <PATH_TO_RESOURCE_ID> --http-method <METHOD>
            --type <INTEGRATION_TYPE> --request-templates <REQUEST_TEMPLATE_MAP>
```

Wobei *<REQUEST_TEMPLATE_MAP>*REQUEST_TEMPLATE_MAP eine Zuordnung vom Inhaltstyp zu einem String der anzuwendenden Vorlage ist. Das Mapping weist folgende Struktur auf:

```
Content_type1=template_string,Content_type2=template_string
```

oder im JSON-Syntax:

```
{"content_type1": "template_string"
```

```
...}
```

Das folgende Beispiel demonstriert, wie Sie mit dem `put-integration-response`-Befehl einen API-Antwortcode überschreiben:

```
aws apigateway put-integration-response --rest-api-id <API_ID> --resource-
id <PATH_TO_RESOURCE_ID> --http-method <METHOD>
            --status-code <STATUS_CODE> --response-templates <RESPONSE_TEMPLATE_MAP>
```

wobei *<RESPONSE_TEMPLATE_MAP>*das gleiche Format wie *<REQUEST_TEMPLATE_MAP>* oben hat.

Beispiel: Überschreiben Sie die Anforderungsparameter und Header einer API mithilfe des SDK für JavaScript

Das folgende Beispiel demonstriert, wie Sie mit dem `put-integration`-Befehl einen Antwortcode überschreiben:

Anfrage:

```
var params = {
  httpMethod: 'STRING_VALUE', /* required */
  resourceId: 'STRING_VALUE', /* required */
  restApiId: 'STRING_VALUE', /* required */
  type: HTTP | AWS | MOCK | HTTP_PROXY | AWS_PROXY, /* required */
  requestTemplates: {
    '<Content_type>': 'TEMPLATE_STRING',
    /* '<String>': ... */
  },
};
apigateway.putIntegration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Antwort:

```
var params = {
  httpMethod: 'STRING_VALUE', /* required */
  resourceId: 'STRING_VALUE', /* required */
  restApiId: 'STRING_VALUE', /* required */
  statusCode: 'STRING_VALUE', /* required */
  responseTemplates: {
```

```
    '<Content_type>': 'TEMPLATE_STRING',
    /* '<String>': ... */
  },
};
apigateway.putIntegrationResponse(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Mappings von Anfrage- und Antwortdaten über die API Gateway-Konsole einrichten

Um die API Gateway-Konsole zur Definition der Integrationsanfrage/-antwort der API zu verwenden, folgen Sie diesen Anweisungen.

> **Note**
>
> Für diese Anleitungen müssen die Schritte in API-Integrationsanfrage über die API Gateway-Konsole einrichten bereits abgeschlossen sein.

1. Wählen Sie im Bereich Ressourcen Ihre Methode aus.
2. Klicken Sie auf der Registerkarte Integrationsanfrage unter Einstellungen für Integrationsanfragen auf Bearbeiten.
3. Wählen Sie eine Option für Request body passthrough, um zu konfigurieren, wie der Methodenanforderungstext eines nicht zugewiesenen Inhaltstyps ohne Transformation an die Lambda-Funktion, den HTTP-Proxy oder den Service-Proxy durch die Integrationsanforderung weitergeleitet wird. AWS Es gibt drei Optionen:

   - Wählen Sie Wenn keine Vorlage mit dem angeforderten Inhaltstyp-Header übereinstimmt aus, wenn der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend weitergereicht werden soll, wenn der Inhaltstyp der Methodenanforderung keiner der den Zuordnungsvorlagen zugeordneten Inhaltstypen entspricht.

     > **Note**
     >
     > Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_MATCH` als `passthroughBehavior`-Eigenschaftswert für die Integration-Ressource festlegen.

- Wählen Sie When there are no templates defined (recommended) aus, wenn der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, falls in der Integrationsanforderung keine Mapping-Vorlage definiert ist. Wenn eine Vorlage definiert und diese Option ausgewählt ist, wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit einer Antwort "HTTP 415 Unsupported Media Type" zurückgewiesen.

  > **Note**
  >
  > Beim Aufruf der API Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_TEMPLATE` als `passthroughBehavior`-Eigenschaftswert für die Integration-Ressource festlegen.

- Wählen Sie Never aus, wenn die Methodenanforderung nicht übergeben werden soll, falls der Inhaltstyp der Methodenanforderung keinem der Inhaltstypen entspricht, die mit den in der Integrationsanforderung definierten Mapping-Vorlagen verknüpft sind, oder falls in der Integrationsanforderung keine Mapping-Vorlage definiert ist. Die Methodenanforderung eines nicht zugeordneten Inhaltstyps wird mit einer Antwort "HTTP 415 Unsupported Media Type" zurückgewiesen.

  > **Note**
  >
  > Beim Aufruf der API Gateway-API wählen Sie diese Option aus, indem Sie `NEVER` als `passthroughBehavior`-Eigenschaftswert für die Integration-Ressource festlegen.

Weitere Informationen zu Integrations-Passthrough-Verhalten finden Sie unter Integrations-Pass-Through-Verhalten.

4. Gehen Sie wie folgt vor, um für einen HTTP-Proxy oder einen AWS Dienstproxy einen Pfadparameter, einen Abfragezeichenfolgenparameter oder einen Header-Parameter, der in der Integrationsanforderung definiert ist, einem entsprechenden Pfadparameter, Abfragezeichenfolgenparameter oder Header-Parameter in der Methodenanforderung des HTTP-Proxys oder AWS Dienstproxys zuzuordnen:

   a. Wählen Sie entweder URL-Pfadparameter, Parameter für URL-Abfragezeichenfolgen oder HTTP-Anforderungs-Header aus und klicken Sie dann entsprechend auf Pfad hinzufügen, Abfragezeichenfolge hinzufügen oder Header hinzufügen.

b. Geben Sie unter Name den Namen des Pfadparameters, des Abfragezeichenfolgenparameters oder des Header-Parameters im HTTP-Proxy oder AWS Dienstproxy ein.

c. Geben Sie unter Zugeordnet von den Zuordnungswert für den Pfadparameter, Abfragezeichenfolge-Parameter oder Header-Parameter ein. Verwenden Sie eines der folgenden Formate:

- **`method.request.path.`*`parameter-name`*** für einen Pfadparameter namens *`parameter-name`*, wie auf der Seite Methodenanforderung definiert.
- **`method.request.querystring.`*`parameter-name`*** für einen Abfragezeichenfolgenparameter namens *`parameter-name`*, wie auf der Seite Methodenanforderung definiert.
- **`method.request.multivaluequerystring.`*`parameter-name`*** für einen mehrwertigen Abfragezeichenfolgenparameter namens *`parameter-name`*, wie auf der Seite Methodenanforderung definiert.
- **`method.request.header.`*`parameter-name`*** für einen Header-Parameter namens *`parameter-name`*, wie auf der Seite Methodenanforderung definiert.

  Alternativ können Sie einen Literalzeichenfolgenwert (in einfachen Anführungszeichen) für einen Integrations-Header festlegen.
- **`method.request.multivalueheader.`*`parameter-name`*** für einen mehrwertigen Header-Parameter namens *`parameter-name`*, wie auf der Seite Methodenanforderung definiert.

d. Klicken Sie auf die Schaltfläche Hinzufügen, wenn Sie weitere Parameter hunzufügen möchten.

5. Wählen Sie Zuordnungsvorlagen aus, um eine Vorlage hinzuzufügen.
6. Wählen Sie Zuordnungsvorlage hinzufügen aus, um eine Zuordnungsvorlage für eine eingehende Anforderung zu definieren. Geben Sie als Inhaltstyp einen Inhaltstyp ein (z. B. **`application/json`**). Geben Sie dann die Zuordnungsvorlage ein. Weitere Informationen finden Sie unter Grundlegendes zu Zuweisungsvorlagen.
7. Wählen Sie Save (Speichern) aus.
8. Sie können eine Integrationsantwort vom Backend einer Methodenantwort der API zuordnen, die an die aufrufende Anwendung zurückgegeben wird. Dies umfasst die Rückkehr von den auf dem Backend verfügbaren zu den vom Client ausgewählten Antwort-Headern und die Umwandlung des Datenformats der Nutzlast der Backend-Antwort in ein von der API vorgegebenes

Format. Sie können diese Zuordnung durch die Konfiguration von Methodenantwort und Integrationsantworten vorgeben.

Gehen Sie wie folgt vor, damit die Methode ein benutzerdefiniertes Antwortdatenformat erhält, das auf dem HTTP-Statuscode basiert, der von der Lambda-Funktion, dem HTTP-Proxy oder dem AWS Dienstproxy zurückgegeben wird:

a. Wählen Sie Integrationsantworten aus. Klicken Sie entweder auf Bearbeiten für Standard - Antwort, um die Einstellungen für einen HTTP-Antwortcode 200 aus der Methode heraus festzulegen. Alternativ klicken Sie auf Antwort erstellen, um die Einstellungen für einen beliebigen anderen HTTP-Antwortstatuscode aus der Methode heraus festzulegen.

b. Geben Sie für Lambda-Fehler-Regex (für eine Lambda-Funktion) oder HTTP-Status-Regex (für einen HTTP-Proxy oder AWS Service-Proxy) einen regulären Ausdruck ein, um anzugeben, welche Lambda-Funktionsfehlerzeichenfolgen (für eine Lambda-Funktion) oder HTTP-Antwortstatuscodes (für einen HTTP-Proxy oder AWS Dienstproxy) dieser Ausgabezuordnung zugeordnet werden. Beispiel: Wenn Sie dieses Ausgabe-Mapping alle 2xx-HTTP-Antwortstatuscodes von einem HTTP-Proxy zuordnen möchten, geben Sie „**`2\d{2}`** für HTTP status regex ein. Wenn Se eine Fehlermeldung mit „Ungültige Anfrage“ von einer Lambda-Funktion an eine `400 Bad Request`-Antwort zurückgegeben werden soll, geben Sie **`.*Invalid request.*`** als den Lambda-Fehler-Regex-Ausdruck ein. Wenn jedoch `400 Bad Request` für alle nicht zugeordneten Lambda-Fehlermeldungen zurückgegeben werden soll, geben Sie **`(\n|.)+`** für Lambda-Fehler-Regex ein. Dieser letzte reguläre Ausdruck kann als Standardfehlerantwort einer API verwendet werden.

> **Note**
>
> API Gateway verwendet für das Antwort-Mapping Regexe-Anweisungen im Java-Pattern-Stil. Weitere Informationen finden Sie unter Pattern in der OracleDokumentation.
> Die Fehlermuster werden mit der gesamten Zeichenfolge der Eigenschaft `errorMessage` in der Lambda-Antwort abgeglichen, die von `callback(errorMessage)` in Node.js oder `throw new MyException(errorMessage)` in Java gefüllt wird. Außerdem werden die Escape-Zeichen geschützter Zeichen entfernt, bevor der reguläre Ausdruck angewendet wird.

Wenn Sie als Auswahlmuster zum Filtern der Antworten verwenden, bedenken Sie, dass eine Antwort mit einem Zeilenumbruch ("\n") möglicherweise nicht zu einer Übereinstimmung führt.

c. Wenn diese Option aktiviert ist, wählen Sie für Methodenantwortstatus den HTTP-Antwortstatuscode aus, den Sie auf der Seite Methodenantwort definiert haben.

d. Geben Sie unter Header-Zuordnungen einen Zuordnungswert für jeden Header ein, den Sie für den HTTP-Antwortstatuscode auf der Seite Methodenantwort definiert haben. Verwenden Sie für den Mapping value eines der folgenden Formate:

- **`integration.response.multivalueheaders.`*`header-name`*** wobei *`header-name`* der Name eines mehrwertigen Antwort-Headers aus dem Backend ist.

  Zum Beispiel: um den `Date`-Header der Backend-Antwort als `Timestamp`-Header einer API-Methode-Antwort zurückzugeben, enthält die Spalte Response header einen Timestamp-Eintrag und der zugehörige Mapping value sollte auf integration.response.multivalueheaders.Date gesetzt sein.

- **`integration.response.header.`*`header-name`*** wobei *`header-name`* der Name eines einzelnen Antwort-Headers aus dem Backend ist.

  Zum Beispiel: um den `Date`-Header der Backend-Antwort als `Timestamp`-Header einer API-Methode-Antwort zurückzugeben, enthält die Spalte Response header einen Timestamp-Eintrag und der zugehörige Mapping value sollte auf integration.response.header.Date gesetzt sein.

e. Wählen Sie Zuordnungsvorlagen aus und klicken Sie dann auf Zuordnungsvorlage hinzufügen. Geben Sie im Feld Inhaltstyp den Inhaltstyp der Daten ein, die von der Lambda-Funktion, dem HTTP-Proxy oder dem AWS Dienstproxy an die Methode übergeben werden. Geben Sie dann die Zuordnungsvorlage ein. Weitere Informationen finden Sie unter Grundlegendes zu Zuweisungsvorlagen.

f. Wählen Sie Save (Speichern) aus.

Beispieldatenmodelle und Zuordnungsvorlagen für API Gateway

Die folgenden Abschnitte enthalten Beispiele für Modelle und Mapping-Vorlagen, die als Ausgangspunkt für Ihre eigenen APIs in API Gateway verwendet werden können. Weitere Informationen zu Datentransformationen finden Sie unter Grundlegendes zu Zuweisungsvorlagen. Weitere Informationen zu Datenmodellen finden Sie unterthe section called “Datenmodelle”.

Themen

- Beispiel für ein Fotoalbum
- Beispiel für Nachrichtenartikel

Beispiel für ein Fotoalbum

Das folgende Beispiel zeigt eine Fotoalbum-API in API Gateway. Wir bieten ein Beispiel für die Datentransformation, zusätzliche Modelle und Zuweisungsvorlagen.

Themen

- Beispiel für eine Datentransformation
- Eingabemodell für Fotodaten
- Ausgabemodell für Fotodaten
- Eingabe-Zuweisungsvorlage für Fotodaten

Beispiel für eine Datentransformation

Das folgende Beispiel zeigt, wie Sie Eingabedaten zu Fotos mithilfe einer VTL-Zuweisungsvorlage (Velocity Template Language) transformieren können. Weitere Informationen zur Velocity Template Language finden Sie in der Apache Velocity – VTL-Reference.

Eingaben

```
{
  "photos": {
    "page": 1,
    "pages": "1234",
    "perpage": 100,
    "total": "123398",
    "photo": [
      {
        "id": "12345678901",
        "owner": "23456789@A12",
        "photographer_first_name" : "Saanvi",
        "photographer_last_name" : "Sarkar",
        "secret": "abc123d456",
        "server": "1234",
        "farm": 1,
        "title": "Sample photo 1",
        "ispublic": true,
```

```
        "isfriend": false,
        "isfamily": false
      },
      {
        "id": "23456789012",
        "owner": "34567890@B23",
        "photographer_first_name" : "Richard",
        "photographer_last_name" : "Roe",
        "secret": "bcd234e567",
        "server": "2345",
        "farm": 2,
        "title": "Sample photo 2",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
      }
    ]
  }
}
```

Ausgab
Z
uweisur
vorlage

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.pho
tographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.ispublic,
      "isfriend": $elem.isfriend,
      "isfamily": $elem.isfamily
    }#if($foreach.hasNext),#end

#end
  ]
}
```

Ausgabdaten

```
{
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    }
  ]
}
```

Eingabemodell für Fotodaten

Sie können ein Modell für Ihre Eingabedaten definieren. Dieses Eingabemodell erfordert, dass Sie ein Foto hochladen, und es spezifiziert mindestens 10 Fotos für jede Seite. Sie können dieses Eingabemodell verwenden, um ein SDK zu generieren oder um eine Anforderungsvalidierung für Ihre API zu aktivieren.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosInputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "object",
      "required" : [
      "photo"
      ],
      "properties": {
        "page": { "type": "integer" },
        "pages": { "type": "string" },
```

```
        "perpage": { "type": "integer", "minimum" : 10 },
        "total": { "type": "string" },
        "photo": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "id": { "type": "string" },
              "owner": { "type": "string" },
              "photographer_first_name" : {"type" : "string"},
              "photographer_last_name" : {"type" : "string"},
              "secret": { "type": "string" },
              "server": { "type": "string" },
              "farm": { "type": "integer" },
              "title": { "type": "string" },
              "ispublic": { "type": "boolean" },
              "isfriend": { "type": "boolean" },
              "isfamily": { "type": "boolean" }
            }
          }
        }
      }
    }
  }
}
```

Ausgabemodell für Fotodaten

Sie können ein Modell für Ihre Ausgabedaten definieren. Sie können dieses Modell für ein Methodenantwortmodell verwenden; dies ist erforderlich, wenn Sie ein stark typisiertes SDK für die API generieren. Dies stellt sicher, dass die Ausgabe an eine geeignete Klasse in Java oder Objective-C übergeben wird.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
```

```
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

Eingabe-Zuweisungsvorlage für Fotodaten

Sie können eine Zuweisungsvorlage definieren, um Eingabedaten zu ändern. Sie können Eingabedaten für weitere Funktionsintegrationen oder Integrationsantworten ändern.

```
#set($inputRoot = $input.path('$'))
{
  "photos": {
    "page": $inputRoot.photos.page,
    "pages": "$inputRoot.photos.pages",
    "perpage": $inputRoot.photos.perpage,
    "total": "$inputRoot.photos.total",
    "photo": [
#foreach($elem in $inputRoot.photos.photo)
      {
        "id": "$elem.id",
        "owner": "$elem.owner",
        "photographer_first_name" : "$elem.photographer_first_name",
        "photographer_last_name" : "$elem.photographer_last_name",
        "secret": "$elem.secret",
        "server": "$elem.server",
        "farm": $elem.farm,
        "title": "$elem.title",
        "ispublic": $elem.ispublic,
        "isfriend": $elem.isfriend,
        "isfamily": $elem.isfamily
      }#if($foreach.hasNext),#end

#end
    ]
  }
```

```
}
```

Beispiel für Nachrichtenartikel

Das folgende Beispiel zeigt eine Nachrichtenartikel-API in API Gateway. Wir bieten ein Beispiel für die Datentransformation, zusätzliche Modelle und Zuweisungsvorlagen.

Themen

- Beispiel für eine Datentransformation
- Eingabemodell für Nachrichtendaten
- Ausgabemodell für Nachrichtendaten
- Eingabe-Mapping-Vorlage für Nachrichtendaten

Beispiel für eine Datentransformation

Das folgende Beispiel zeigt, wie Sie Eingabedaten zu einem Nachrichtenartikel mithilfe einer Velocity Template Language (VTL) -Mapping-Vorlage transformieren können. Weitere Informationen zur Velocity Template Language finden Sie in der Apache Velocity – VTL-Reference.

Eingabedaten

```
{
  "count": 1,
  "items": [
    {
      "last_updated_date": "2015-04-24",
      "expire_date": "2016-04-25",
      "author_first_name": "John",
      "description": "Sample Description",
      "creation_date": "2015-04-20",
      "title": "Sample Title",
      "allow_comment": true,
      "author": {
        "last_name": "Doe",
        "email": "johndoe@example.com",
        "first_name": "John"
      },
      "body": "Sample Body",
      "publish_date": "2015-04-25",
      "version": "1",
      "author_last_name": "Doe",
```

```
      "parent_id": 2345678901,
      "article_url": "http://www.example.com/articles/3456789012"
    }
  ],
  "version": 1
}
```

Ausgab
Z
uweisur
vorlage

```
#set($inputRoot = $input.path('$'))
{
  "count": $inputRoot.count,
  "items": [
#foreach($elem in $inputRoot.items)
    {
      "creation_date": "$elem.creation_date",
      "title": "$elem.title",
      "author": "$elem.author.first_name $elem.author.last_name",
      "body": "$elem.body",
      "publish_date": "$elem.publish_date",
      "article_url": "$elem.article_url"
    }#if($foreach.hasNext),#end

#end
  ],
  "version": $inputRoot.version
}
```

Ausgab
ten

```
{
  "count": 1,
  "items": [
    {
      "creation_date": "2015-04-20",
      "title": "Sample Title",
      "author": "John Doe",
      "body": "Sample Body",
      "publish_date": "2015-04-25",
      "article_url": "http://www.example.com/articles/3456789012"
    }
  ],
  "version": 1
}
```

Eingabemodell für Nachrichtendaten

Sie können ein Modell für Ihre Eingabedaten definieren. Dieses Eingabemodell erfordert, dass ein Nachrichtenartikel eine URL, einen Titel und einen Hauptteil enthält. Sie können dieses Eingabemodell verwenden, um ein SDK zu generieren oder um eine Anforderungsvalidierung für Ihre API zu aktivieren.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "NewsArticleInputModel",
  "type": "object",
  "properties": {
    "count": { "type": "integer" },
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "required": [
      "article_url",
      "title",
      "body"
      ],
        "properties": {
          "last_updated_date": { "type": "string" },
          "expire_date": { "type": "string" },
          "author_first_name": { "type": "string" },
          "description": { "type": "string" },
          "creation_date": { "type": "string" },
          "title": { "type": "string" },
          "allow_comment": { "type": "boolean" },
          "author": {
            "type": "object",
            "properties": {
              "last_name": { "type": "string" },
              "email": { "type": "string" },
              "first_name": { "type": "string" }
            }
          },
          "body": { "type": "string" },
          "publish_date": { "type": "string" },
          "version": { "type": "string" },
          "author_last_name": { "type": "string" },
          "parent_id": { "type": "integer" },
```

```
          "article_url": { "type": "string" }
        }
      }
    },
    "version": { "type": "integer" }
  }
}
```

Ausgabemodell für Nachrichtendaten

Sie können ein Modell für Ihre Ausgabedaten definieren. Sie können dieses Modell für ein Methodenantwortmodell verwenden; dies ist erforderlich, wenn Sie ein stark typisiertes SDK für die API generieren. Dies stellt sicher, dass die Ausgabe an eine geeignete Klasse in Java oder Objective-C übergeben wird.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

Eingabe-Mapping-Vorlage für Nachrichtendaten

Sie können eine Zuweisungsvorlage definieren, um Eingabedaten zu ändern. Sie können Eingabedaten für weitere Funktionsintegrationen oder Integrationsantworten ändern.

```
#set($inputRoot = $input.path('$'))
{
  "count": $inputRoot.count,
  "items": [
#foreach($elem in $inputRoot.items)
    {
      "last_updated_date": "$elem.last_updated_date",
      "expire_date": "$elem.expire_date",
      "author_first_name": "$elem.author_first_name",
      "description": "$elem.description",
      "creation_date": "$elem.creation_date",
      "title": "$elem.title",
      "allow_comment": "$elem.allow_comment",
      "author": {
        "last_name": "$elem.author.last_name",
        "email": "$elem.author.email",
        "first_name": "$elem.author.first_name"
      },
      "body": "$elem.body",
      "publish_date": "$elem.publish_date",
      "version": "$elem.version",
      "author_last_name": "$elem.author_last_name",
      "parent_id": $elem.parent_id,
      "article_url": "$elem.article_url"
    }#if($foreach.hasNext),#end

#end
  ],
  "version": $inputRoot.version
}
```

Daten-Mapping-Referenz für Amazon API Gateway-API-Anfragen und Antworten

In diesem Abschnitt wird die Einrichtung von Daten-Mappings von Methodenanforderungsdaten einer API – einschließlich anderer, in `context`-, `stage`- oder `util`-Variablen gespeicherter Daten – zu den entsprechenden Integrationsanforderungs-Parametern und von Integrationsantwortdaten – einschließlich der anderen Daten – zu den Methodenantwortparametern beschrieben. Die Methodenanforderungsdaten umfassen Anforderungsparameter (Pfad, Abfragezeichenfolge, Header) und den Text. Die Integrationsantwortdaten enthalten Antwortparameter (Header) und den Text. Weitere Informationen zur Verwendung von "stage"-Variablen finden Sie unter Referenz für Amazon API Gateway-Stufenvariablen.

Themen

Zuweisen von Methodenanforderungsdaten zu Integrationsanforderungs-Parametern

Integrationsanforderungs-Parameter in Form von Pfadvariablen, Abfragezeichenfolgen oder Headern können aus allen definierten Methodenanforderungs-Parametern und der Nutzlast zugewiesen werden.

In der folgenden Tabelle *PARAM_NAME* ist der Name eines Methodenanforderungs-Parameters des angegebenen Parametertyps. Er muss dem regulären Ausdruck `'^[a-zA-Z0-9._$-]+$]'` entsprechen. Es muss definiert worden sein, bevor es referenziert werden kann. *JSONPath_EXPRESSION* ist ein JSONPath-Ausdruck für ein JSON-Feld des Hauptteils einer Anforderung oder Antwort.

> **Note**
>
> Das `"$"` ist in dieser Syntax ausgelassen.

Zuweisungsausdrücke für Integrationsanforderungsdaten

Zugewiesene Datenquelle	Zuweisungsausdruck
Methodenanforderungspfad	`method.request.path.` *PARAM_NAME*
Abfragezeichenfolge der Methodenanforderung	`method.request.querystring.` *PARAM_NAME*
Mehrwertmethode Abfrage Abfrage Zeichenfolge	`method.request.multivaluequerystring.` *PARAM_NAME*
Methodenanforderungs-Header	`method.request.header.` *PARAM_NAME*

Zugewiesene Datenquelle	Zuweisungsausdruck
Mehrfachmethodenanforderungs-Header	`method.request.multivalueheader.` *PARAM_NAME*
Methodenanforderungstext	`method.request.body`
Hauptteil der Methodenanforderung (JsonPath)	`method.request.body.` *JSONPath_EXPRESSION* .
Stufenvariablen	`stageVariables.` *VARIABLE_NAME*
Kontextvariablen	`context.`*VARIABLE_NAME* , wobei die Variable zu den unterstützten Kontextvariablen gehören muss.
Statischer Wert	*'STATIC_VALUE'* . *STATIC_VALUE* ist ein Zeichenfolgenliteral, das in einfache Anführungszeichen eingeschlossen werden muss.

Example Zuweisen von Methoden-Mapping-Parametern in OpenAPI

Das folgende Beispiel zeigt einen OpenAPI-Ausschnitt, der:

- den Header der Methodenanforderung mit dem Namen `methodRequestHeaderParam` in den Pfad des Integration Request-Parameters`integrationPathParam` zuweist
- den Multi-Wert der Abfragezeichenfolge der Methodenanforderung mit Namen `methodRequestQueryParam`, der Abfragezeichenfolge der Integrationsanforderung namens `integrationQueryParam`zuweist

```
...
"requestParameters" : {

    "integration.request.path.integrationPathParam" :
 "method.request.header.methodRequestHeaderParam",
    "integration.request.querystring.integrationQueryParam" :
 "method.request.multivaluequerystring.methodRequestQueryParam"
```

```
}
...
```

Integrationsanforderungs-Parameter können mithilfe eines JSONPath-Ausdrucks auch von Feldern im JSON-Anforderungstext zugewiesen werden. Die folgende Tabelle zeigt die Mapping-Ausdrücke für einen Methodenanforderungstext und dessen JSON-Felder.

Example Mapping vom Methodenanforderungstext in OpenAPI

Das folgende Beispiel zeigt einen OpenAPI-Ausschnitt, der 1) den Methodenanforderungstext dem Integrationsanforderungs-Header namens `body-header` und 2) ein JSON-Feld des Texts, wie durch einen JSON-Ausdruck dargestellt (`petstore.pets[0].name`, ohne das `$.`-Präfix) zuweist.

```
...
"requestParameters" : {

    "integration.request.header.body-header" : "method.request.body",
    "integration.request.path.pet-name" : "method.request.body.petstore.pets[0].name",

}
...
```

Zuweisen von Integrationsantwortdaten zu Methodenantwort-Headern

Methodenantwort-Header-Parameter können aus allen Integrationsantwort-Headern oder Integrationsantworttexten, `$context`-Variablen oder statischen Werten zugewiesen werden.

Mapping-Ausdrücke für Methodenantwort-Header

Zugewiesene Datenquelle	Mapping-Ausdruck
Integrationsantwort-Header	`integration.response.header.` *PARAM_NAME*
Integrationsantwort-Header	`integration.response.multivalueheader.` *PARAM_NAME*

Zugewiesene Datenquelle	Mapping-Ausdruck
Integrationsantworttext	`integration.response.body`
Antwortstelle für die Integration (JsonPath)	`integration.respon se.body.` *`JSONPath_EXPRESSION`*
Stufenvariable	`stageVariables.` *`VARIABLE_NAME`*
Kontextvariable	`context.`*`VARIABLE_NAME`* , wobei die Variable zu den unterstützten Kontextvariablen gehören muss.
Statischer Wert	*`'STATIC_VALUE'`* . *`STATIC_VALUE`* ist ein Zeichenfolgenliteral, das in einfache Anführungszeichen eingeschlossen werden muss.

Example Daten-Mapping aus Integrationsantworten in OpenAPI

Das folgende Beispiel zeigt einen OpenAPI-Ausschnitt, der 1) das `redirect.url`-"JSONPath"-Feld der Integrationsantwort dem `location`-Header der Anforderungsantwort und 2) den `x-app-id`-Header der Integrationsantwort dem `id`-Header der Methodenantwort zuweist.

```
...
"responseParameters" : {

    "method.response.header.location" : "integration.response.body.redirect.url",
    "method.response.header.id" : "integration.response.header.x-app-id",
    "method.response.header.items" : "integration.response.multivalueheader.item",

}
...
```

Zuweisen von Anforderungs- und Antwortnutzlasten zwischen Methode und Integration

Amazon API Gateway nutzt das VTL-Modul (Velocity Template Language) zur Verarbeitung von Mapping-Vorlagen in Texten für die Integrationsanforderung und Integrationsantwort.

Die Mapping-Vorlagen übersetzen Methodenanforderungsnutzlasten in die entsprechenden Integrationsanforderungsnutzlasten und Integrationsantworttexte in Methodenantworttexte.

Die VTL-Vorlagen verwenden JSONPath-Ausdrücke, andere Parameter wie Aufrufkontexte und Stufenvariablen sowie Hilfsprogrammfunktionen, um die JSON-Daten zu verarbeiten.

Wenn ein Modell zur Beschreibung der Datenstruktur einer Nutzlast definiert ist, kann Amazon API Gateway anhand dieses Modells ein Mapping-Vorlagenskelett für eine Integrationsanforderung oder -antwort erstellen. Sie können das Vorlagenskelett als Hilfe bei der Anpassung und Erweiterung des VTL-Mapping-Skripts verwenden. Sie können aber auch eine vollständig neue Mapping-Vorlage erstellen, ohne ein Modell für die Datenstruktur der Nutzlast zu definieren.

Auswählen einer VTL-Mapping-Vorlage

Amazon API Gateway verwendet die folgende Logik zur Auswahl einer Mapping-Vorlage in Velocity Template Language (VTL), um den Payload aus einer Methodenanforderung der entsprechenden Integrationsanforderung oder die Nutzlast aus einer Integrationsantwort der entsprechenden Methodenantwort zuzuweisen.

Für einen Anfrage-Payload verwendet API Gateway den `Content-Type`-Header-Wert der Anfrage als Schlüssel zur Auswahl der Mapping-Vorlage für den Anfrage-Payload. Bei Payload-Antworten verwendet API Gateway den `Accept`-Header-Wert der eingehenden Anfrage als Schlüssel für die Auswahl der Mapping-Vorlage für den Payload.

Wenn der `Content-Type`-Header nicht in der Anfrage vorhanden ist, nimmt API Gateway an, dass sein Standardwert `application/json` ist. Für eine solche Anfrage verwendet API Gateway `application/json` als Standardschlüssel zur Auswahl der Mapping-Vorlage, falls eine solche definiert ist. Wenn keine Vorlage mit diesem Schlüssel übereinstimmt, übergibt API Gateway die Nutzlast ohne Zuweisung, wenn die Eigenschaft passthroughBehavior auf `WHEN_NO_MATCH` oder `WHEN_NO_TEMPLATES` eingestellt ist.

Wenn der `Accept`-Header nicht in der Anfrage angegeben ist, geht API Gateway davon aus, dass sein Standardwert `application/json` ist. In diesem Fall wählt API Gateway eine vorhandene Mapping-Vorlage für `application/json` aus, um den Payload der Antwort zuzuordnen. Wenn keine Vorlage für `application/json` definiert ist, wählt API Gateway die erste vorhandene Vorlage aus und verwendet sie als Standard für das Mapping des Payloads der Antwort. Genauso verwendet API Gateway die erste vorhandene Vorlage, wenn der angegebene `Accept`-Header-Wert nicht mit einem vorhandenen Vorlagenschlüssel übereinstimmt. Wenn keine Vorlage definiert ist, leitet API Gateway den Payload der Antwort einfach nicht zugeordnet weiter.

Angenommen, eine API verfügt über eine für eine Anforderungsnutzlast definierte `application/json`-Vorlage und eine für die Antwortnutzlast definierte `application/xml`-Vorlage. Wenn der Client die `"Content-Type : application/json"`- und `"Accept : application/xml"`-Header in der Anforderung festlegt, werden sowohl die Anforderungs- als auch die Antwortnutzlasten mit den entsprechenden Mapping-Vorlagen verarbeitet. Wenn der `Accept:application/xml`-Header fehlt, wird die `application/xml`-Mapping-Vorlage für die Zuweisung der Antwortnutzlast verwendet. Wenn Sie stattdessen eine Antwortnutzlast ohne Zuweisung zurückgeben möchten, richten Sie eine leere Vorlage für ei `application/json`.

Die `Accept`- und `Content-Type`-Header verwenden bei der Auswahl der Mapping-Vorlage nur den MIME-Typ. Beispiel: Für einen `"Content-Type: application/json; charset=UTF-8"`-Header wird eine Anforderungsvorlage mit dem `application/json`-Schlüssel ausgewählt.

Integrations-Pass-Through-Verhalten

Wenn bei Nicht-Proxyintegrationen eine Methodenanforderung eine Nutzlast enthält und entweder der `Content-Type`-Header nicht der angegebenen Mapping-Vorlage entspricht oder keine Mapping-Vorlage definiert ist, können Sie die vom Client bereitgestellte Anforderungsnutzlast ohne Transformation über die Integrationsanforderung an das Backend weiterleiten. Dieser Prozess wird als "Integrations-Pass-Through" bezeichnet.

Bei Proxy-Integrationen leitet API Gateway die gesamte Anfrage an Ihr Backend weiter. Sie haben keine Möglichkeit, das Weiterleitungsverhalten zu ändern.

Das tatsächliche Pass-Through-Verhalten einer eingehenden Anforderung basiert auf der Option, die Sie beim Einrichten der Integrationsanforderung für eine bestimmte Mapping-Vorlage ausgewählt haben, und dem Content-Type-Header, den der Client in der eingehenden Anforderung festgelegt hat. Es gibt drei Optionen:

Wenn keine Vorlage mit dem angeforderten Inhaltstyp-Header übereinstimmt

> Wählen Sie diese Option, falls der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern der Inhaltstyp der Methodenanforderung keinem mit den Zuweisungsvorlagen verknüpften Inhaltstyp entspricht.
>
> Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_MATCH` als `passthroughBehavior`-Eigenschaftswert für die Integration festlegen.

Wenn keine Vorlagen definiert sind (empfohlen)

Wählen Sie diese Option, falls der Methodenanforderungstext ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern in der Integrationsanforderung keine Zuweisungsvorlage definiert ist. Wenn eine Vorlage definiert und diese Option ausgewählt ist, wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit einer Antwort "HTTP 415 Unsupported Media Type" zurückgewiesen.

Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `WHEN_NO_TEMPLATES` als `passthroughBehavior`-Eigenschaftswert für die Integration festlegen.

Niemals

Wählen Sie diese Option, falls der Methodenanforderungstext nicht ohne Umwandlung durch die Integrationsanforderung an das Backend übergeben werden soll, sofern in der Integrationsanforderung keine Zuweisungsvorlage definiert ist. Wenn eine Vorlage definiert und diese Option ausgewählt ist, wird die Methodenanforderung eines nicht zugeordneten Inhaltstyps mit einer Antwort "HTTP 415 Unsupported Media Type" zurückgewiesen.

Beim Aufruf der API-Gateway-API wählen Sie diese Option aus, indem Sie `NEVER` als `passthroughBehavior`-Eigenschaftswert für die Integration festlegen.

Die folgenden Beispiele veranschaulichen das mögliche Pass-Through-Verhalten.

Beispiel 1: Eine Mapping-Vorlage wird in der Integrationsanforderung für den Content-Type `application/json` definiert.

Content-Type-Header\ausgewählte Pass-Through-Option	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
Keine (standardmäßig `application/json`)	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.	Die Anforderungsnutzlast wird anhand der Vorlage umgewandelt.
`application/json`	Die Anforderungsnutzlast wird	Die Anforderungsnutzlast wird	Die Anforderungsnutzlast wird

Content-Type-Header\ausgewählte Pass-Through-Option	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
	anhand der Vorlage umgewandelt.	anhand der Vorlage umgewandelt.	anhand der Vorlage umgewandelt.
`application/xml`	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.

Beispiel 2: Eine Mapping-Vorlage wird in der Integrationsanforderung für den Content-Type `application/xml` definiert.

Content-Type-Header\ausgewählte Pass-Through-Option	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
Keine (Standard `application/json`)	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.
`application/json`	Die Anforderungsnutzlast wird nicht umgewandelt, sondern unverändert an das Backend gesendet.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.	Die Anforderung wird mit der HTTP-Antwort `415 Unsupported Media Type` abgelehnt.
`application/xml`	Die Anforderungsnutzlast wird	Die Anforderungsnutzlast wird	Die Anforderungsnutzlast wird

Content-Type-Header\ausgewählte Pass-Through-Option	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
	anhand der Vorlage umgewandelt.	anhand der Vorlage umgewandelt.	anhand der Vorlage umgewandelt.

Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen

Dieser Abschnitt enthält Referenzinformationen für die Variablen und Funktionen, die Amazon API Gateway für die Verwendung mit Datenmodellen, Autorisierern, Zuordnungsvorlagen und CloudWatch Zugriffsprotokollierung definiert. Detaillierte Informationen zur Verwendung dieser Variablen und Funktionen finden Sie unter Grundlegendes zu Zuweisungsvorlagen. Weitere Informationen zur Velocity Template Language (VTL) finden Sie in der VTL-Referenz.

Themen

- $contextVariablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch
- Beispiel für $context-Variablenvorlage
- $context-Variablen nur für Zugriffsprotokollierung
- $input-Variablen
- Beispiele für $input-Variablenvorlage
- $stageVariables
- $util-Variablen

> **Note**
>
> Weitere Informationen zu `$method`- und `$integration`-Variablen finden Sie unter the section called “Anforderungs- und Antwortdaten-Mapping – Referenz”.

`$context`Variablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch

Die folgenden `$context` Variablen können in Datenmodellen, Autorisierern, Zuordnungsvorlagen und der Zugriffsprotokollierung verwendet werden. CloudWatch API Gateway fügt möglicherweise zusätzliche Kontextvariablen hinzu.

Informationen zu `$context` Variablen, die nur für die CloudWatch Zugriffsprotokollierung verwendet werden können, finden Sie unterthe section called “`$context`-Variablen nur für Zugriffsprotokollierung”.

Parameter	Beschreibung
`$context.accountId`	Die AWS Konto-ID des API-Besitzers.
`$context.apiId`	Die ID, die API Gateway Ihrer API zuweist.
`$context.authorizer.claims.`*`property`*	Eine Eigenschaft der Claims, die aus dem Amazon Cognito-Benutzerpool zurückgeg eben werden, nachdem der Methodenaufrufer erfolgreich authentifiziert wurde. Weitere Informationen finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”. **Note** Bei einem Aufruf von `$context.authorizer.claims` wird null (0) zurückgegeben.
`$context.authorizer.principalId`	Die ID des Prinzipalbenutzers in Verbindung mit dem Token, das vom Client gesendet und von einem API Gateway-Lambda-Genehmiger (früher als benutzerdefinierter Genehmige r bekannt) zurückgegeben wurde. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden”.

Parameter	Beschreibung
`$context.authorizer.` *`property`*	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der `context`-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende `context`-Zuweisung zurück: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` Dann gibt der Aufruf von `$context.authorizer.key` die Zeichenfolge `"value"` zurück, der Aufruf von `$context.authorizer.numKey` gibt die Zeichenfolge `"1"` zurück und der Aufruf von `$context.authorizer.boolKey` gibt den Wert `"true"` zurück. Weitere Informationen finden Sie unter the section called “Lambda-Genehmiger verwenden”.
`$context.awsEndpointRequestId`	Die Anforderungs-ID des AWS Endpunkts.
`$context.deploymentId`	Die ID der API-Bereitstellung.
`$context.domainName`	Der zum Aufrufen der API verwendete vollständige Domänennamen. Dieser Wert sollte mit dem für den eingehenden `Host`-Header übereinstimmen.

Parameter	Beschreibung
`$context.domainPrefix`	Das erste Label der `$context.domainName` .
`$context.error.message`	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält. Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponseBody-Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und für die Zugriffsprotokollierung verwendet werden. Weitere Informationen finden Sie unter the section called “Metriken” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen”.
`$context.error.messageString`	Die Wert von `$context.error.message` in Anführungszeichen, d. h. `"$context.error.message"` .
`$context.error.responseType`	Ein Typ von. GatewayResponse Diese Variable kann nur für die einfache Variablenersetzung in einer GatewayResponseBody-Mapping-Vorlage, die nicht von der Velocity Template Language-Engine verarbeitet wird, und für die Zugriffsprotokollierung verwendet werden. Weitere Informationen finden Sie unter the section called “Metriken” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen”.
`$context.error.validationErrorString`	Eine Zeichenfolge mit einer detaillierten Validierungs-Fehlermeldung.

Parameter	Beschreibung
`$context.extendedRequestId`	Die erweiterte ID, die API Gateway generiert und der API-Anfrage zuweist. Die erweiterte Anforderungs-ID enthält zusätzliche nützliche Informationen für Debugging und Fehlerbeh ebung.
`$context.httpMethod`	Die verwendete HTTP-Methode. Gültige Werte sind: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` und `PUT`.
`$context.identity.accountId`	Die der AWS Anfrage zugeordnete Konto-ID.
`$context.identity.apiKey`	Bei API-Methoden, für die ein API-Schlüssel erforderlich ist, ist diese Variable der API-Schlüssel für die Methodenanforderung. Bei Methoden, für die kein API-Schlüssel erforderl ich ist, ist diese Variable nichtig. Weitere Informationen finden Sie unter the section called “Nutzungspläne”.
`$context.identity.apiKeyId`	Die API-Schlüssel-ID für die API-Anforderung, falls ein API-Schlüssel erforderlich ist.
`$context.identity.caller`	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Ressource n unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Beschreibung
`$context.identity.cognitoAuthenticationProvider`	Eine durch Komma getrennte Liste der Amazon Cognito-Authentifizierungsanbieter, die vom Aufrufer, der die Anfrage stellt, verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Weitere Informationen finden Sie unter Verbundidentitäten verwenden im Amazon Cognito-Entwicklerhandbuch.
`$context.identity.cognitoAuthenticationType`	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind `authenticated` für authentifizierte Identitäten und `unauthenticated` für nicht authentifizierte Identitäten.
`$context.identity.cognitoIdentityId`	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
`$context.identity.cognitoIdentityPoolId`	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Beschreibung
`$context.identity.principalOrgId`	Die AWS -Organisations-ID.
`$context.identity.sourceIp`	Die Quell-IP-Adresse der unmittelbaren TCP-Verbindung, die die Anfrage an den API-Gateway-Endpunkt stellt.
`$context.identity.clientCert.clientCertPem`	Das PEM-codierte Clientzertifikat, das der Client während der gegenseitigen TLS-Authentifizierung präsentiert hat. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
`$context.identity.clientCert.subjectDN`	Der Distinguished Name des Zertifikatantragsstellers, den ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domain-Namens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
`$context.identity.clientCert.issuerDN`	Der Distinguished Name des Ausstellers des Zertifikats, das ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.

Parameter	Beschreibung
`$context.identity.clientCert.serialNumber`	Die Seriennummer des Zertifikats. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
`$context.identity.clientCert.validity.notBefore`	Das Datum, vor dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
`$context.identity.clientCert.validity.notAfter`	Das Datum, nach dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift. Nur in Zugriffsprotokollen vorhanden, wenn die gegenseitige TLS-Authentifizierung fehlschlägt.
`$context.identity.vpcId`	Die VPC-ID der VPC, die die Anfrage an den API-Gateway-Endpunkt stellt.
`$context.identity.vpceId`	Die VPC-Endpunkt-ID des VPC-Endpunkts, der die Anfrage an den API-Gateway-Endpunkt stellt. Nur vorhanden, wenn Sie über eine private API verfügen.
`$context.identity.user`	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Ressourcen unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Beschreibung
`$context.identity.userAgent`	Die `User-Agent` -Kopfzeile des API-Aufrufers.
`$context.identity.userArn`	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.isCanaryRequest`	Gibt zurück`true`, ob die Anfrage an den Canary gerichtet war und `false` ob die Anfrage nicht an den Canary gerichtet war. Nur vorhanden, wenn Sie einen Canary aktiviert haben.
`$context.path`	Der Anforderungspfad. Bei einer Nicht-Proxy-Anforderungs-URL von `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` lautet der `$context.path` -Wert beispielsweise `/{stage}/root/child` .
`$context.protocol`	Das Anforderungsprotokoll ist z. B, `HTTP/1.1`. **Note** API-Gateway-APIs können HTTP/2-Anfragen akzeptieren, aber API Gateway sendet Anfragen mithilfe von HTTP/1.1 an Backend-Integrationen. Infolgedessen wird das Anforderungsprotokoll als HTTP/1.1 protokolliert, auch wenn ein Client eine Anfrage sendet, die HTTP/2 verwendet.

Parameter	Beschreibung
`$context.requestId`	Eine ID für die Anforderung. Clients können diese Anforderungs-ID überschreiben. Verwenden von `$context.extendedRequestId` für eine eindeutige Anforderungs-ID, die API Gateway generiert.
`$context.requestOverride.header.` *`header_name`*	Der Anforderungs-Header-Override. Wenn dieser Parameter definiert ist, enthält er die Header, die statt der HTTP Header, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.
`$context.requestOverride.path.` *`path_name`*	Der Anforderungspfad-Override. Wenn dieser Parameter definiert ist, enthält er den Anforderungspfad, der statt der URL-Pfadparameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden soll. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.
`$context.requestOverride.querystring.` *`querystring_name`*	Der Abfragestring-Override. Wenn dieser Parameter definiert ist, enthält er die Abfragestrings, die statt der URL-Abfragestring-Parameter, die im Bereich Integrationsanforderung definiert sind, verwendet werden sollen. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.

Parameter	Beschreibung
`$context.responseOverride.header. header_name`	Der Antwort-Header-Override. Wenn dieser Parameter definiert ist, enthält er den Header, der anstelle des Antwort-Headers, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.
`$context.responseOverride.status`	Der Antwortstatuscode-Override. Wenn dieser Parameter definiert ist, enthält er den Statuscode, der anstelle des Methoden-Antwortstatus, der als Standard-Mapping im Bereich Integrationsantwort definiert ist, ausgegeben werden soll. Weitere Informationen finden Sie unter Verwendung einer Mapping-Vorlage zum Überschreiben der Anforderungs- und Antwortparameter und -Statuscodes einer API.
`$context.requestTime`	Die Anforderungszeit im CLF-Format (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Die Anforderungszeit im Epoch-Format in Millisekunden.
`$context.resourceId`	Der Bezeichner, den API Gateway Ihrer Ressource zuweist.

Parameter	Beschreibung
`$context.resourcePath`	Der Pfad zu Ihrer Ressource. Beim Nicht-Proxy-Anforderungs-URI von `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` lautet der `$context.resourcePath` -Wert beispielsweise `/root/child` . Weitere Informationen finden Sie unter Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen.
`$context.stage`	Die Bereitstellungsstufe der API-Anforderung (z. B. `Beta` oder `Prod`).
`$context.wafResponseCode`	Die von AWS WAF empfangene Antwort: `WAF_ALLOW` oder `WAF_BLOCK` . Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called “AWS WAF”.
`$context.webaclArn`	Vollständiger ARN der Web-Zugriffskontrollliste (Web-ACL), anhand deren entschieden wird, ob die Anforderung zugelassen oder blockiert wird. Wird nicht festgelegt, wenn die Stufe mit keiner Web-ACL verknüpft ist. Weitere Informationen finden Sie unter the section called “AWS WAF”.

Beispiel für **`$context`**-Variablenvorlage

Die Verwendung einer `$context`-Variablen in einer Mapping-Vorlage kann sinnvoll sein, wenn Ihre API-Methode strukturierte Daten an ein Backend übermittelt, das ein bestimmtes Datenformat erfordert.

Das folgende Beispiel zeigt eine Mapping-Vorlage für die Zuordnung eingehender `$context` Variablen zu Backend-Variablen mit geringfügig unterschiedlichen Namen in der Nutzlast einer Integrationsanforderung:

> **Note**
>
> Eine der Variablen ist ein API-Schlüssel. In diesem Beispiel wird davon ausgegangen, dass die Methode einen API-Schlüssel benötigt.

```
{
    "stage" : "$context.stage",
    "request_id" : "$context.requestId",
    "api_id" : "$context.apiId",
    "resource_path" : "$context.resourcePath",
    "resource_id" : "$context.resourceId",
    "http_method" : "$context.httpMethod",
    "source_ip" : "$context.identity.sourceIp",
    "user-agent" : "$context.identity.userAgent",
    "account_id" : "$context.identity.accountId",
    "api_key" : "$context.identity.apiKey",
    "caller" : "$context.identity.caller",
    "user" : "$context.identity.user",
    "user_arn" : "$context.identity.userArn"
}
```

Die Ausgabe dieser Mapping-Vorlage sollte wie folgt aussehen:

```
{
  stage: 'prod',
  request_id: 'abcdefg-000-000-0000-abcdefg',
  api_id: 'abcd1234',
  resource_path: '/',
  resource_id: 'efg567',
  http_method: 'GET',
  source_ip: '192.0.2.1',
  user-agent: 'curl/7.84.0',
  account_id: '111122223333',
  api_key: 'MyTestKey',
  caller: 'ABCD-0000-12345',
  user: 'ABCD-0000-12345',
  user_arn: 'arn:aws:sts::111122223333:assumed-role/Admin/carlos-salazar'
}
```

$context-Variablen nur für Zugriffsprotokollierung

Die folgenden $context-Variablen sind nur für Zugriffsprotokollierung verfügbar. Weitere Informationen finden Sie unter the section called “CloudWatch Logs”. (WebSocket APIs finden Sie unterthe section called “Metriken”.)

Parameter	Beschreibung
$context.authorize.error	Die Autorisierungsfehlermeldung.
$context.authorize.latency	Die Autorisierungslatenz in ms
$context.authorize.status	Der Statuscode, der von einem Autorisierungsversuch zurückgegeben wurde.
$context.authorizer.error	Die von einem Genehmiger zurückgegebene Fehlermeldung.
$context.authorizer.integrationLatency	Der Genehmiger-Latenz in ms.
$context.authorizer.integrationStatus	Der von einem Lambda-Genehmiger zurückgegebene Statuscode.
$context.authorizer.latency	Der Genehmiger-Latenz in ms.
$context.authorizer.requestId	Die Anforderungs-ID des AWS Endpunkts.
$context.authorizer.status	Der von einem Genehmiger zurückgegebene Statuscode.
$context.authenticate.error	Die von einem Authentifizierungsversuch zurückgegebene Fehlermeldung.
$context.authenticate.latency	Die Authentifizierungslatenz in ms
$context.authenticate.status	Der Statuscode, der von einem Authentifizierungsversuch zurückgegeben wurde.
$context.customDomain.basePathMatched	Der Pfad für ein API-Mapping, mit dem eine eingehende Anforderung übereinstimmte.

Parameter	Beschreibung
	Gilt, wenn ein Client einen benutzerdefinierten Domain-Namen für den Zugriff auf eine API verwendet. Wenn ein Client beispielsweise eine Anforderung an `https://api.example.com/v1/orders/1234` sendet und die Anforderung dem API-Mapping mit dem Pfad `v1/orders` übereinstimmt, lautet der Wert `v1/orders`. Weitere Informationen hierzu finden Sie unter the section called “API-Mappings”.
`$context.endpointType`	Der Endpunkttyp der API.
`$context.integration.error`	Die von einer Integration zurückgegebene Fehlermeldung.
`$context.integration.integrationStatus`	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.
`$context.integration.latency`	Die Integrationslatenz in Millisekunden. Äquivalent mit `$context.integrationLatency`.
`$context.integration.requestId`	Die AWS Anforderungs-ID des Endpunkts. Äquivalent mit `$context.awsEndpointRequestId`.
`$context.integration.status`	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird.
`$context.integrationLatency`	Die Integrationslatenz in Millisekunden.

Parameter	Beschreibung
`$context.integrationStatus`	Für die Lambda-Proxyintegration stellt dieser Parameter den Statuscode dar, der vom Lambda-Funktionscode zurückgegeben wurde AWS Lambda, nicht aus dem Back-End-Lambda-Funktionscode.
`$context.responseLatency`	Die Antwortlatenz in Millisekunden.
`$context.responseLength`	Die Länge der Antwortnutzlast in Byte.
`$context.status`	Der Status der Methodenantwort.
`$context.waf.error`	Die Fehlermeldung wurde von zurückgegeben. AWS WAF
`$context.waf.latency`	Die AWS WAF Latenz in ms.
`$context.waf.status`	Der Statuscode wurde von zurückgegeben AWS WAF.
`$context.xrayTraceId`	Die Trace-ID für die X-Ray-Trace. Weitere Informationen finden Sie unter the section called “Einrichten AWS X-Ray”.

`$input`-Variablen

Die `$input`-Variable stellt die Nutzlast und die Parameter der Methodenanforderung dar, die von der Mapping-Vorlage verarbeitet werden sollen. Er bietet die folgenden Funktionen:

Variable und Funktion	Beschreibung
`$input.body`	Gibt die Nutzlast der Raw-Anforderung als Zeichenfolge zurück.
`$input.json(x)`	Diese Funktion wertet einen JSONPath-Ausdruck aus und gibt die Ergebnisse als JSON-Zeichenfolge zurück.

Variable und Funktion	Beschreibung
	Beispielsweise gibt `$input.json('$.pets')` eine JSON-Zeichenfolge zurück, die die `pets`-Struktur abbildet. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.
`$input.params()`	Gibt die Zuweisung aller Anforderungsparameter zurück. Wir empfehlen, das Ergebnis `$util.escapeJavaScript` zu bereinigen, um einen möglichen Injektionsangriff zu vermeiden. Um die vollständige Kontrolle über die Bereinigung von Anfragen zu erhalten, verwenden Sie eine Proxy-Integration ohne Vorlage und übernehmen Sie die Anforderungsbereinigung in Ihrer Integration.
`$input.params(x)`	Gibt aus der Zeichenfolge eines Parameternamens `x` aus dem Pfad, der Abfragezeichenfolge oder dem Header-Wert (in dieser Reihenfolge) den Wert eines Methodenanforderungs-Parameters zurück. Wir empfehlen, den Parameter `$util.escapeJavaScript` zu bereinigen, um einen möglichen Injektionsangriff zu vermeiden. Um die vollständige Kontrolle über die Parameterbereinigung zu erhalten, verwenden Sie eine Proxy-Integration ohne Vorlage und übernehmen Sie die Anforderungsbereinigung in Ihrer Integration.

Variable und Funktion	Beschreibung
`$input.path(x)`	Empfängt eine JSONPath-Ausdruckszeichenfolge (`x`) und gibt eine JSON-Objektdarstellung des Ergebnisses zurück. Dies ermöglicht einen nativen Zugriff auf Elemente der Nutzlast in Apache Velocity Template Language (VTL) und deren Bearbeitung. Beispiel: Der Ausdruck `$input.path('$.pets')` könnte das folgende Objekt zurückgeben: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` gibt `"3"` zurück. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.

Beispiele für **`$input`**-Variablenvorlage

Die folgenden Beispiele zeigen, wie die `$input` Variablen in Mapping-Vorlagen verwendet werden. Sie können eine Scheinintegration oder eine Lambda-Integration ohne Proxy verwenden, die das Eingabeereignis zurück an API Gateway zurückgibt, um diese Beispiele auszuprobieren.

Beispiel für Parameter-Mapping-Vorlage

Im folgenden Beispiel werden alle Anforderungsparameter, einschließlich, und `path` `querystringheader`, über eine JSON-Nutzlast an den Integrationsendpunkt übergeben:

```
#set($allParams = $input.params())
{
  "params" : {
    #foreach($type in $allParams.keySet())
    #set($params = $allParams.get($type))
    "$type" : {
      #foreach($paramName in $params.keySet())
      "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
      #if($foreach.hasNext),#end
      #end
    }
    #if($foreach.hasNext),#end
    #end
  }
}
```

Für eine Anfrage, die die folgenden Eingabeparameter enthält:

- Ein Pfadparameter mit dem Namen `myparam`
- Zeichenkettenparameter abfragen `querystring1=value1,value2&querystring2=value3`
- Überschriften `"header1" : "value1""header2" : "value2","header3" : "value3"`.

Die Ausgabe dieser Mapping-Vorlage sollte wie folgt aussehen:

```
{
  "params" : {
        "path" : {
            "path" : "myparam"
                }
    ,        "querystring" : {
```

```
            "querystring1" : "value1,value2"
      ,           "querystring2" : "value3"
               }
    ,         "header" : {
            "header3" : "value3"
      ,           "header2" : "value2"
      ,           "header1" : "value1"
               }
          }
}
```

Beispiel für eine JSON-Mapping-Vorlage

Sie können die `$input`-Variable verwenden, um Abfragezeichenfolgen und Anforderungstext mit oder ohne die Verwendung von Modellen abzurufen. Möglicherweise möchten Sie auch den Parameter und die Payload oder einen Unterabschnitt der Payload abrufen. Die folgenden drei Beispiele zeigen, wie das geht.

Im folgenden Beispiel wird eine Zuordnungsvorlage verwendet, um einen Unterabschnitt der Nutzlast abzurufen. In diesem Beispiel werden der Eingabeparameter `name` und dann der gesamte POST-Text abgerufen:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$')
}
```

Für eine Anfrage, die die Parameter der Abfragezeichenfolge `name=Bella&type=dog` und den folgenden Hauptteil enthält:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

Die Ausgabe dieser Zuordnungsvorlage sollte wie folgt aussehen:

```
{
    "name" : "Bella",
    "body" : {"Price":"249.99","Age":"6"}
}
```

Wenn die JSON-Eingabe Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können JavaScript, gibt API Gateway möglicherweise eine 400-Antwort zurück. `$util.escapeJavaScript($input.json('$'))`Anwenden, um sicherzustellen, dass die JSON-Eingabe ordnungsgemäß analysiert werden kann.

Das vorherige Beispiel mit `$util.escapeJavaScript($input.json('$'))` Applied lautet wie folgt:

```
{
    "name" : "$input.params('name')",
    "body" : $util.escapeJavaScript($input.json('$'))
}
```

In diesem Fall sollte die Ausgabe dieser Mapping-Vorlage wie folgt aussehen:

```
{
    "name" : "Bella",
    "body": {\"Price\":\"249.99\",\"Age\":\"6\"}
}
```

Beispiel für einen JSONPath-Ausdruck

Das folgende Beispiel zeigt, wie ein JSONPath-Ausdruck an die Methode `json()` übergeben wird. Sie könnten auch einen Unterabschnitt Ihres Anfragetextobjekts lesen, indem Sie einen Punkt,, verwenden`.`, um eine Eigenschaft anzugeben:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$.Age')
}
```

Für eine Anfrage, die die Parameter der Abfragezeichenfolge `name=Bella&type=dog` und den folgenden Hauptteil enthält:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

Die Ausgabe dieser Zuordnungsvorlage sollte wie folgt aussehen:

```
{
    "name" : "Bella",
    "body" : "6"
}
```

Wenn die Payload einer Methodenanforderung Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können JavaScript, gibt API Gateway möglicherweise eine Antwort zurück. `400` `$util.escapeJavaScript()`Anwenden, um sicherzustellen, dass die JSON-Eingabe ordnungsgemäß analysiert werden kann.

Das vorherige Beispiel mit `$util.escapeJavaScript($input.json('$.Age'))` Applied lautet wie folgt:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

In diesem Fall sollte die Ausgabe dieser Mapping-Vorlage wie folgt aussehen:

```
{
    "name" : "Bella",
    "body": "\"6\""
}
```

Beispiel für Anfrage und Antwort

Im folgenden Beispiel wird `$input.params()``$input.path()`, und `$input.json()` für eine Ressource mit dem Pfad verwendet`/things/{id}`:

```
{
    "id" : "$input.params('id')",
    "count" : "$input.path('$.things').size()",
    "things" : $input.json('$.things')"
}
```

Für eine Anfrage, die den Pfadparameter `123` und den folgenden Hauptteil enthält:

```
{
      "things": {
            "1": {},
```

```
            "2": {},
            "3": {}
      }
}
```

Die Ausgabe dieser Mapping-Vorlage sollte wie folgt aussehen:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Wenn die Payload einer Methodenanforderung Zeichen ohne Escape-Zeichen enthält, die nicht analysiert werden können JavaScript, gibt API Gateway möglicherweise eine Antwort zurück. `400` `$util.escapeJavaScript()`Anwenden, um sicherzustellen, dass die JSON-Eingabe ordnungsgemäß analysiert werden kann.

Das vorherige Beispiel mit `$util.escapeJavaScript($input.json('$.things'))` Applied lautet wie folgt:

```
{
     "id" : "$input.params('id')",
     "count" : "$input.path('$.things').size()",
     "things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

Die Ausgabe dieser Mapping-Vorlage sollte wie folgt aussehen:

```
{"id":"123","count":"3","things":"{\"1\":{},\"2\":{},\"3\":{}}"}
```

Weitere Mapping-Beispiele finden Sie unter Grundlegendes zu Zuweisungsvorlagen.

`$stageVariables`

Stufenvariablen können zur Zuweisung von Parametern und für Mapping-Vorlagen verwendet werden sowie als Platzhalter in ARNs und URLs bei Methodenintegrationen. Weitere Informationen finden Sie unter the section called “Einrichten von Stufenvariablen”.

Syntax	Beschreibung
`$stageVariables.` *`<variable_name>`* , `$stageVariables['` *`<variable`*	*`<variable_name>`* variable name gibt einen Stufenvariablennamen an.

Syntax	Beschreibung
_name> '] oder ${stageVariables[' *<variable_name>* ']}	

$util-Variablen

Die $util-Variable enthält Dienstprogrammfunktionen, die in Mapping-Vorlagen verwendet werden.

> **Note**
>
> Sofern nicht anders angegeben, wird UTF-8 als Standardzeichensatz genutzt.

Funktion	Beschreibung
$util.escapeJavaScript()	Escapiert die Zeichen in einer Zeichenfolge mithilfe von JavaScript Zeichenfolgenregeln. **Note** Mit dieser Funktion werden alle einfachen Anführungszeichen (') durch Escape-Zeichen (\') geschützt. Allerdings sind diese durch Escape-Zeichen geschützten einfachen Anführungszeichen in JSON nicht zulässig. Sofern die Ausgabe dieser Funktion in einer JSON-Eigenschaft verwendet werden soll, müssen alle einfachen Anführungszeichen, die durch Escape-Zeichen geschützt sind (\'), wieder in reguläre einfache Anführungszeichen (') geändert werden. Das wird im folgenden Beispiel veranschaulicht:

Funktion	Beschreibung
	<pre>"input" : "$util.escapeJavaScript(data).replaceAll("\\'",'''')"</pre>

`$util.parseJson()`	Erhält das "stringify"-JSON-Objekt und gibt eine Objektdarstellung des Ergebnisses zurück. Mit dem Ergebnis dieser Funktion können Sie Elemente der Nutzlast, die in Apache Velocity Template Language (VTL) sind, aufrufen und bearbeiten. Angenommen, Sie haben folgende Nutzlast:
	<pre>{"errorMessage":"{\"key1\":\"var1\",\"key2\":{\"arr\":[1,2,3]}}"}</pre>Und verwenden die folgende Mapping-Vorlage:
	<pre>#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))
	{
	   "errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]
	}</pre>Dann erhalten Sie die folgende Ausgabe:
	<pre>{
	   "errorMessageObjKey2ArrVal" : 1
	}</pre>

`$util.urlEncode()`	Konvertiert eine Zeichenfolge in das Format „application/x-www-form-urlencoded“.

Funktion	Beschreibung
`$util.urlDecode()`	Dekodiert eine Zeichenfolge „application/x-www-form-urlencoded“.
`$util.base64Encode()`	Codiert die Daten in eine base64-verschlüsselte Zeichenfolge.
`$util.base64Decode()`	Decodiert die Daten einer base64-verschlüsselten Zeichenfolge.

Gateway-Antworten in API Gateway

Eine Gateway-Antwort wird durch einen Antworttyp identifiziert, der durch API Gateway definiert ist. Die Antwort besteht aus einem HTTP-Statuscode, eine Reihe zusätzlicher Header, die von Parameter-Mappings angegeben werden, und einer Nutzlast, die durch eine andere als eine VTL-Mapping-Vorlage generiert wird.

In der API Gateway REST API wird eine Gateway-Antwort durch die dargestellt GatewayResponse. In OpenAPI wird eine `GatewayResponse` Instanz durch die Erweiterung x-amazon-apigateway-gateway-responses.GatewayResponse beschrieben.

Zum Aktivieren einer Gateway-Antwort richten Sie sie für einen unterstützten Antworttyp auf API-Ebene ein. Wenn API Gateway eine Antwort dieses Typs zurückgibt, werden die in der Gateway-Antwort definierten Header-Mappings und Nutzlast-Mapping-Vorlagen angewendet, um die zugewiesenen Ergebnisse an den API-Aufrufer zurückzugeben.

Im folgenden Abschnitt wird gezeigt, wie Gateway-Antworten mit Hilfe der API Gateway-Konsole und der API Gateway-REST-API eingerichtet werden.

Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen

Wenn API Gateway eine eingehende Anfrage nicht verarbeiten kann, wird eine Fehlermeldung an den Client gesendet, ohne die Anfrage an das Integrations-Backend weiterzuleiten. Standardmäßig enthält diese Meldung eine kurze Beschreibung des Fehlers. Wenn Sie beispielsweise versuchen, einen Vorgang für eine nicht definierte API-Ressource aufzurufen, erhalten Sie eine Fehlermeldung mit folgendem Inhalt: `{ "message": "Missing Authentication Token" }`. Wenn Sie API Gateway noch nicht kennen, ist die Ursache eine Problems möglicherweise schwer zu ermitteln.

Bei einigen Fehlerantworten ermöglicht API Gateway die Anpassung durch API-Entwickler. So können Antworten in verschiedenen Formaten zurückzugeben werden. Bei dem Beispiel `Missing Authentication Token` können Sie der ursprünglichen Antwortnutzlast einen Hinweis auf die Fehlerursache hinzufügen, wie in diesem Beispiel: `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`.

Wenn Ihre API zwischen einer externen Börse und der AWS Cloud vermittelt, verwenden Sie VTL-Mapping-Vorlagen für Integrationsanfragen oder Integrationsantworten, um die Nutzlast von einem Format einem anderen Format zuzuordnen. Die VTL-Mapping-Vorlagen funktionieren aber nur für gültige Anforderungen mit erfolgreichen Antworten.

Bei ungültigen Anfragen umgeht API Gateway die Integration vollständig und gibt eine Fehlerantwort zurück. Sie müssen die Anpassung für die Fehlermeldungen nutzen, um diese in einem austauschkonformen Format darzustellen. Hier wird die Anpassung in einer anderen als der VTL-Mapping-Vorlage dargestellt, die nur einfache Variablenersetzungen unterstützt.

Wir bezeichnen die von API Gateway generierte Fehlerantwort auf alle von API Gateway generierten Antworten allgemein als Gateway-Antworten. Dies unterscheidet die von API Gateway generierten Antworten von den Integrationsantworten. Eine Mapping-Vorlage für eine Gateway-Antwort kann auf `$context`-Variablenwerte und `$stageVariables`-Eigenschaftswerte sowie auf Methodenanforderungs-Parameter im Format `method.request.`*`param-position`*`.`*`param-name`* zugreifen.

Weitere Informationen zu `$context`-Variablen finden Sie unter `$context`Variablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch . Mehr über `$stageVariables` erfahren Sie unter `$stageVariables`. Weitere Informationen zu Methodenanforderungs-Parametern finden Sie unter the section called “`$input`-Variablen”.

Themen

- Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole
- Gateway-Antwort mit der API Gateway REST-API einrichten
- Einrichten der Gateway-Antwortanpassung in OpenAPI
- Gateway-Antworttypen

Einrichten einer Gateway-Antwort für eine REST-API mit der API-Gateway-Konsole

So passen Sie eine Gateway-Antwort über die API Gateway-Konsole an:

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Gateway-Antworten.
4. Wählen Sie einen Antworttyp und dann Bearbeiten aus. In dieser Anleitung verwenden wir Fehlendes Authentifizierungstoken als Beispiel.
5. Sie können den von API Gateway generierten Statuscode ändern, um einen anderen Statuscode auszugeben, der den Anforderungen Ihrer API entspricht. In diesem Beispiel ändert die Anpassung den Statuscode vom Standard (`403`) zu `404`, da diese Fehlermeldung auftritt, wenn ein Client eine nicht unterstützte oder ungültige Ressource aufruft, die als nicht gefunden betrachtet werden kann.
6. Um zu den benutzerdefinierten Headern zurückzukehren, wählen Sie Header hinzufügen hinzufügen unter Antwort-Header aus. Zur Veranschaulichung fügen wir die folgenden benutzerdefinierten Header hinzu:

   ```
   Access-Control-Allow-Origin:'a.b.c'
   x-request-id:method.request.header.x-amzn-RequestId
   x-request-path:method.request.path.petId
   x-request-query:method.request.querystring.q
   ```

   In den vorstehenden Header-Mappings wird dem `'a.b.c'`-Header ein statischer Domänenname (`Allow-Control-Allow-Origin`) hinzugefügt, um den CORS-Zugriff auf die API zuzulassen; der `x-amzn-RequestId` in der Antwort wird der Eingabeanforderungs-Header `request-id` zugewiesen; dem `petId`-Header in der Antwort wird die `request-path`-Pfadvariable zugewiesen und dem q-Header der Antwort wird der `request-query`-Abfrageparameter der ursprünglichen Anforderung zugewiesen.
7. Behalten Sie unter Antwortvorlagen die Option `application/json` für Inhaltstyp bei und geben Sie die folgende Text-Mapping-Vorlage in den Editor Vorlagentext ein:

   ```
   {
        "message":"$context.error.messageString",
        "type": "$context.error.responseType",
        "statusCode": "'404'",
   ```

```
    "stage": "$context.stage",
    "resourcePath": "$context.resourcePath",
    "stageVariables.a": "$stageVariables.a"
}
```

In diesem Beispiel sehen Sie, wie die `$context`- und `$stageVariables`-Eigenschaften den Eigenschaften des Gateway-Antworttexts zugewiesen werden.

8. Wählen Sie Änderungen speichern aus.
9. Stellen Sie die API in einer neuen oder vorhandenen Stufe bereit.

Testen Sie Ihre Gateway-Antwort, indem Sie den folgenden CURL-Befehl aufrufen. Dabei wird davon ausgegangen, dass die Aufruf-URL der entsprechenden API-Methode `https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/{petId}` ist:

```
curl -v -H 'x-amzn-RequestId:123344566' https://o81lxisefl.execute-api.us-
east-1.amazonaws.com/custErr/pets/5/type?q=1
```

Da der zusätzliche Abfragezeichenfolge-Parameter q=1 nicht mit der API kompatibel ist, wird ein Fehler zurückgegeben, um die angegebene Gateway-Antwort auszulösen. Sie sollten eine Gateway-Antwort wie die folgende erhalten:

```
> GET /custErr/pets/5?q=1 HTTP/1.1
Host: o81lxisefl.execute-api.us-east-1.amazonaws.com
User-Agent: curl/7.51.0
Accept: */*

HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: 334
Connection: keep-alive
Date: Tue, 02 May 2017 03:15:47 GMT
x-amzn-RequestId: 123344566
Access-Control-Allow-Origin: a.b.c
x-amzn-ErrorType: MissingAuthenticationTokenException
header-1: static
x-request-query: 1
x-request-path: 5
X-Cache: Error from cloudfront
Via: 1.1 441811a054e8d055b893175754efd0c3.cloudfront.net (CloudFront)
```

```
X-Amz-Cf-Id: nNDR-fX4csbRoAgtQJ16u0rTDz9FZWT-Mk93KgoxnfzDlTUh3flmzA==

{
     "message":"Missing Authentication Token",
     "type": MISSING_AUTHENTICATION_TOKEN,
     "statusCode": '404',
     "stage": custErr,
     "resourcePath": /pets/{petId},
     "stageVariables.a": a
}
```

Im vorherigen Beispiel wird davon ausgegangen, dass das API-Backend Pet Store ist und die API über die definierte Stufenvariable a verfügt.

Gateway-Antwort mit der API Gateway REST-API einrichten

Bevor Sie eine Gateway-Antwort mit der API Gateway-REST-API anpassen können, müssen Sie eine API erstellt und deren ID abgerufen haben. Zum Abrufen der API-ID können Sie der restapi:gateway-responses-Link-Beziehung folgen und das Ergebnis untersuchen.

So passen Sie eine Gateway-Antwort mit der API Gateway-REST-API an:

1. Um eine gesamte GatewayResponseInstanz zu überschreiben, rufen Sie die Aktion gatewayresponse:put auf. Geben Sie im URL-Pfadparameter einen gewünschten responseType an und stellen Sie in der Anforderungsnutzlast die Mappings statusCode, responseParameters und responseTemplates bereit.
2. Um einen Teil einer `GatewayResponse`-Instance zu aktualisieren, rufen Sie die Aktion gatewayresponse:update auf. Geben Sie einen gewünschten `responseType` im URL-Pfadparameter an und stellen Sie in der Anfragenutzlast die einzelnen gewünschten `GatewayResponse`-Eigenschaften bereit, z. B. das Mapping `responseParameters` oder `responseTemplates`.

Einrichten der Gateway-Antwortanpassung in OpenAPI

Mit der `x-amazon-apigateway-gateway-responses`-Erweiterung auf der API-Stammebene können Sie Gateway-Antworten in OpenAPI anpassen. Die folgende OpenAPI-Definition zeigt ein Beispiel für die Anpassung GatewayResponsedes `MISSING_AUTHENTICATION_TOKEN` Typs.

```
  "x-amazon-apigateway-gateway-responses": {
    "MISSING_AUTHENTICATION_TOKEN": {
```

```
      "statusCode": 404,
      "responseParameters": {
        "gatewayresponse.header.x-request-path": "method.input.params.petId",
        "gatewayresponse.header.x-request-query": "method.input.params.q",
        "gatewayresponse.header.Access-Control-Allow-Origin": "'a.b.c'",
        "gatewayresponse.header.x-request-header": "method.input.params.Accept"
      },
      "responseTemplates": {
        "application/json": "{\n     \"message\": $context.error.messageString,\n
    \"type\":  \"$context.error.responseType\",\n     \"stage\":  \"$context.stage
\",\n     \"resourcePath\":  \"$context.resourcePath\",\n     \"stageVariables.a\":
 \"$stageVariables.a\",\n     \"statusCode\": \"'404'\"\n}"
      }
    }
```

In diesem Beispiel wird durch die Anpassung der Standardstatuscode `403` in `404` geändert. Außerdem werden der Gateway-Antwort vier Header-Parameter und eine Text-Mapping-Vorlage für den `application/json`-Medientyp hinzugefügt.

Gateway-Antworttypen

API Gateway stellt die folgenden Gateway-Antworten zur Anpassung durch API-Entwickler zur Verfügung.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
`ACCESS_DENIED`	`403`	Die Gateway-Antwort bei Autorisierungsfehlern, beispielsweise bei einer Verweigerung des Zugriffs durch einen benutzerdefinierten oder Amazon Cognito-Genehmiger. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_4XX` für diese Antwort verwendet.
`API_CONFIGURATION_ERROR`	`500`	Die Gateway-Antwort für eine ungültige API-Konfiguration,

Gateway-Antworttyp	Standardstatuscode	Beschreibung
		einschließlich wenn eine ungültige Endpunktadresse übermittelt wird, wenn die base64-Decodierung von Binärdaten bei Verwendung der binären Unterstützung fehlschlägt oder wenn das Integrationsantwort-Mapping keiner Vorlage entspricht und keine Standardvorlage konfiguriert ist. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_5XX` für diese Antwort verwendet.
`AUTHORIZER_CONFIGURATION_ERROR`	`500`	Die Gateway-Antwort für eine fehlgeschlagene Verbindung zu einem benutzerdefinierten oder Amazon Cognito-Genehmiger. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_5XX` für diese Antwort verwendet.
`AUTHORIZER_FAILURE`	`500`	Die Gateway-Antwort, wenn ein benutzerdefinierter oder Amazon Cognito-Genehmiger den Aufrufer nicht authentifizieren konnte. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_5XX` für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
BAD_REQUEST_PARAMETERS	400	Die Gateway-Antwort, wenn der Anforderungsparameter nicht gemäß einem aktivierten Anforderungsvalidierer überprüft werden kann. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
BAD_REQUEST_BODY	400	Die Gateway-Antwort, wenn der Anforderungstext nicht gemäß einem aktivierten Anforderungsvalidierer überprüft werden kann. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
DEFAULT_4XX	Null	Die Standard-Gateway-Antwort für einen nicht angegebenen Antworttyp mit dem Statuscode 4XX. Durch eine Änderung des Statuscodes dieser Fallback-Gateway-Antwort werden die Statuscodes von allen anderen 4XX-Antworten in den neuen Wert geändert. Mit einem Zurücksetzen des Statuscodes auf null werden die Statuscodes von sämtlichen anderen 4XX-Antworten auf ihre ursprünglichen Werte zurückgesetzt. **Note** AWS WAF benutzerdefinierte Antworten haben Vorrang vor benutzerdefinierten Gateway-Antworten.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
DEFAULT_5XX	Null	Die Standard-Gateway-Antwort für einen nicht angegebenen Antworttyp mit dem Statuscode 5XX. Durch eine Änderung des Statuscodes dieser Fallback-Gateway-Antwort werden die Statuscodes von allen anderen 5XX-Antworten in den neuen Wert geändert. Mit einem Zurücksetzen des Statuscodes auf null werden die Statuscodes von sämtlichen anderen 5XX-Antworten auf ihre ursprünglichen Werte zurückgesetzt.
EXPIRED_TOKEN	403	Fehler bei der Gateway-Antwort auf ein AWS Authentifizierungstoken ist abgelaufen. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
INTEGRATION_FAILURE	504	Die Gateway-Antwort bei einem Integrationsfehler. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_5XX für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
INTEGRATION_TIMEOUT	504	Die Gateway-Antwort, wenn das Zeitlimit für die Integrati on überschritten wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_5XX für diese Antwort verwendet.
INVALID_API_KEY	403	Die Gateway-Antwort, wenn für eine Methode, die einen API-Schlüssel erfordert, ein ungültiger API-Schlüssel übermittelt wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
INVALID_SIGNATURE	403	Die Gateway-Antwort auf einen ungültigen AWS Signaturfehler. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
MISSING_AUTHENTICATION_TOKEN	403	Die Gateway-Antwort, wenn das Authentifizierungs-Token fehlt. Dazu zählen auch die Fälle, bei denen der Client versucht, eine nicht unterstützte API-Methode oder -Ressource aufzurufen. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
QUOTA_EXCEEDED	429	Die Gateway-Antwort, wenn das Kontingent für den Nutzungsplan überschritten wurde. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
REQUEST_TOO_LARGE	413	Die Gateway-Antwort, wenn die Anforderung zu groß ist. Wenn der Antworttyp nicht angegeben ist, ist diese Antwort standardmäßig:. HTTP content length exceeded 10485760 bytes

Gateway-Antworttyp	Standardstatuscode	Beschreibung
RESOURCE_NOT_FOUND	404	Die Gateway-Antwort, wenn Amazon API Gateway die angegebene Ressource nicht finden kann, nachdem eine API-Anfrage die Authentif izierung und Autorisierung – mit Ausnahme der API-Schlü ssel-Authentifizierung und -Autorisierung – durchlaufen hat. Wenn kein Antworttyp angegeben ist, wird standardm äßig DEFAULT_4XX für diese Antwort verwendet.
THROTTLED	429	Die Gateway-Antwort, wenn Drosselungslimits auf Nutzungsplan-, Methoden-, Stufen- oder Kontoebene überschritten wurden. Wenn kein Antworttyp angegeben ist, wird standardmäßig DEFAULT_4XX für diese Antwort verwendet.
UNAUTHORIZED	401	Die Gateway-Antwort, wenn der benutzerdefinierte oder Amazon Cognito-Genehmiger den Aufrufer nicht authentif izieren konnte.

Gateway-Antworttyp	Standardstatuscode	Beschreibung
`UNSUPPORTED_MEDIA_TYPE`	`415`	Die Gateway-Antwort, wenn strenges Passthrough-Verhalten aktiviert ist und eine Nutzlast einen nicht unterstützten Medientyp aufweist. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_4XX` für diese Antwort verwendet.
`WAF_FILTERED`	`403`	Die Gateway-Antwort, wenn eine Anforderung von AWS WAF blockiert wird. Wenn kein Antworttyp angegeben ist, wird standardmäßig `DEFAULT_4XX` für diese Antwort verwendet. **Note** AWS WAF benutzerdefinierte Antworten haben Vorrang vor benutzerdefinierten Gateway-Antworten.

Aktivieren von CORS für eine REST-API-Ressource

Cross-Origin Resource Sharing (CORS) ist eine Browser-Sicherheitsfunktion, die Cross-Origin-HTTP-Anfragen einschränkt, die von Skripts eingeleitet werden, die im Browser ausgeführt werden. Weitere Informationen finden Sie unter Was ist CORS? .

Feststellung, ob die CORS-Unterstützung aktiviert werden soll

Cross-Origin-HTTP-Anfragen zielen auf Folgendes:

- Eine andere Domäne (z. B. von `example.com` an `amazondomains.com`)
- Eine andere Subdomäne (z. B. von `example.com` an `petstore.example.com`)
- Ein anderer Port (z. B. von `example.com` an `example.com:10777`)
- Ein anderes Protokoll (z. B. von `https://example.com` an `http://example.com`)

Wenn Sie nicht auf Ihre API zugreifen können und eine Fehlermeldung erhalten, die `Cross-Origin Request Blocked` enthält, müssen Sie möglicherweise CORS aktivieren.

Cross-Origin-HTTP-Anfragen lassen sich in zwei Arten unterteilen: einfache und nicht einfache Anfragen.

Aktivieren von CORS für eine einfache Anfrage

Eine HTTP-Anfrage ist einfach, wenn alle der folgenden Bedingungen erfüllt sind:

- Sie wird für eine API-Ressource gestellt, die nur `GET`-, `HEAD`- und `POST`-Anfragen erlaubt.
- Wenn es sich um eine `POST`-Methodenanfrage handelt, muss sie einen `Origin`-Header enthalten.
- Der Anfragenutzlast-Inhaltstyp ist `text/plain`, `multipart/form-data` oder `application/x-www-form-urlencoded`.
- Die Anfrage enthält keine benutzerdefinierten Header.
- Alle weiteren Anforderungen, die in der Mozilla CORS-Dokumentation für einfache Anfragen aufgelistet sind.

Bei einfachen quellenübergreifenden `POST`-Methodenanfragen muss die Antwort Ihrer Ressource den Header `Access-Control-Allow-Origin: '*'` oder `Access-Control-Allow-Origin:'origin'` enthalten.

Alle anderen Cross-Origin-HTTP-Anfragen sind nicht einfache Anfragen.

Aktivieren von CORS für eine nicht einfache Anfrage

Wenn die Ressourcen Ihrer API nicht einfache Anfragen erhalten, müssen Sie je nach Integrationstyp zusätzliche CORS-Unterstützung aktivieren.

Aktivieren von CORS für Integrationen ohne Proxy

Für diese Integrationen erfordert das CORS-Protokoll, dass der Browser eine Preflight-Anfrage an den Server sendet und auf die Genehmigung (oder Anforderung der Anmeldeinformationen) vom

Server wartet, bevor die tatsächliche Anfrage gesendet wird. Sie müssen Ihre API so konfigurieren, dass sie eine entsprechende Antwort auf die Preflight-Anfrage sendet.

So erstellen Sie eine Preflight-Antwort:

1. Erstellen Sie eine `OPTIONS`-Methode mit einer Mock-Integration.
2. Fügen Sie der Antwort der Methode 200 die folgenden Antwort-Header hinzu:
   - `Access-Control-Allow-Headers`
   - `Access-Control-Allow-Methods`
   - `Access-Control-Allow-Origin`
3. Stellen Sie das Passthrough-Verhalten der Integration auf ein`NEVER`. In diesem Fall wird die Methodenanforderung eines Inhaltstyps ohne Zuordnung mit der Antwort HTTP 415 Unsupported Media Type zurückgewiesen. Weitere Informationen finden Sie unter Integrations-Pass-Through-Verhalten.
4. Geben Sie Werte für die Antwort-Header ein. Verwenden Sie die folgenden Header-Werte, um alle Ursprünge, alle Methoden und allgemeine Header zuzulassen:
   - `Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'`
   - `Access-Control-Allow-Methods: '*'`
   - `Access-Control-Allow-Origin: '*'`

Nachdem Sie die Preflight-Anfrage erstellt haben, müssen Sie den `Access-Control-Allow-Origin: '*'`- oder `Access-Control-Allow-Origin:`*`'origin'`*-OR-Header für alle CORS-fähigen Methoden für mindestens alle 200 Antworten zurückgeben.

Aktivierung von CORS für Nicht-Proxy-Integrationen mit dem AWS Management Console

Sie können den verwenden, um CORS AWS Management Console zu aktivieren. API Gateway erstellt eine `OPTIONS`-Methode und fügt den `Access-Control-Allow-Origin`-Header zu Ihren vorhandenen Methodenintegrationsantworten hinzu. Dies funktioniert nicht immer und manchmal müssen Sie die Integrationsantwort manuell ändern, um den `Access-Control-Allow-Origin`-Header für alle CORS-fähigen Methoden für mindestens alle 200 Antworten zurückzugeben.

Aktivieren der CORS-Unterstützung für Proxy-Integrationen

Bei einer Lambda-Proxy-Integration oder einer HTTP-Proxy-Integration ist Ihr Backend für die Rückgabe der Header `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods` und `Access-Control-Allow-Headers` verantwortlich, da eine Proxy-Integration keine Integrationsantwort zurückgibt.

Die folgenden Beispiel-Lambda-Funktionen geben die erforderlichen CORS-Header zurück:

Node.js

```
export const handler = async (event) => {
    const response = {
        statusCode: 200,
        headers: {
            "Access-Control-Allow-Headers" : "Content-Type",
            "Access-Control-Allow-Origin": "https://www.example.com",
            "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
        },
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Python 3

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://www.example.com',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET'
        },
        'body': json.dumps('Hello from Lambda!')
    }
```

Themen

- CORS für eine Ressource über die API Gateway-Konsole aktivieren

- CORS für eine Ressource über die API Gateway-Import-API aktivieren
- Testen von CORS

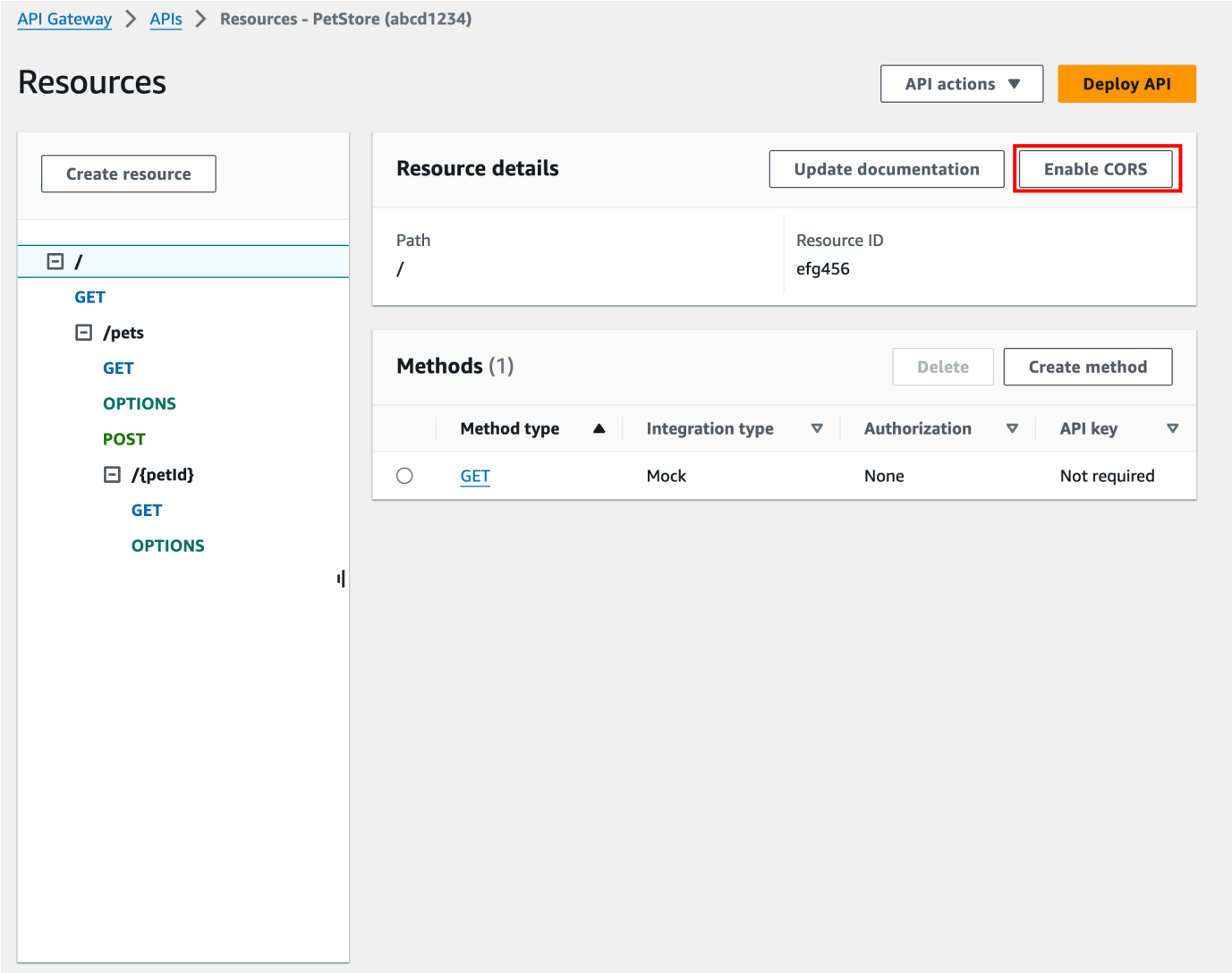
CORS für eine Ressource über die API Gateway-Konsole aktivieren

Sie können die API Gateway-Konsole verwenden, um die CORS-Unterstützung für eine oder alle Methoden einer von Ihnen erstellten REST-API-Ressource zu aktivieren. Nachdem Sie die COR-Unterstützung aktiviert haben, legen Sie das Passthrough-Verhalten für die Integration auf fest. `NEVER` In diesem Fall wird die Methodenanforderung eines Inhaltstyps ohne Zuordnung mit der Antwort HTTP 415 Unsupported Media Type zurückgewiesen. Weitere Informationen finden Sie unter Integrations-Pass-Through-Verhalten.

> **Important**
>
> Ressourcen können untergeordnete Ressourcen enthalten. Die Aktivierung der CORS-Unterstützung für eine Ressource und ihre Methoden führt nicht dazu, dass sie rekursiv für untergeordnete Ressourcen und ihre Methoden aktiviert wird.

So aktivieren Sie die CORS-Unterstützung für eine REST-API-Ressource:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine API aus.
3. Wählen Sie eine Ressource unter Resources.
4. Wählen Sie im Abschnitt Ressourcendetails die Option CORS aktivieren aus.



5. Gehen Sie im Feld CORS aktivieren wie folgt vor:

   a. (Optional) Wenn Sie eine benutzerdefinierte Gateway-Antwort erstellt haben und die CORS-Unterstützung für eine Antwort aktivieren möchten, wählen Sie eine Gateway-Antwort aus.

   b. Wählen Sie die einzelnen Methoden aus, um die CORS-Unterstützung zu aktivieren. Für die `OPTION`-Methode muss CORS aktiviert sein.

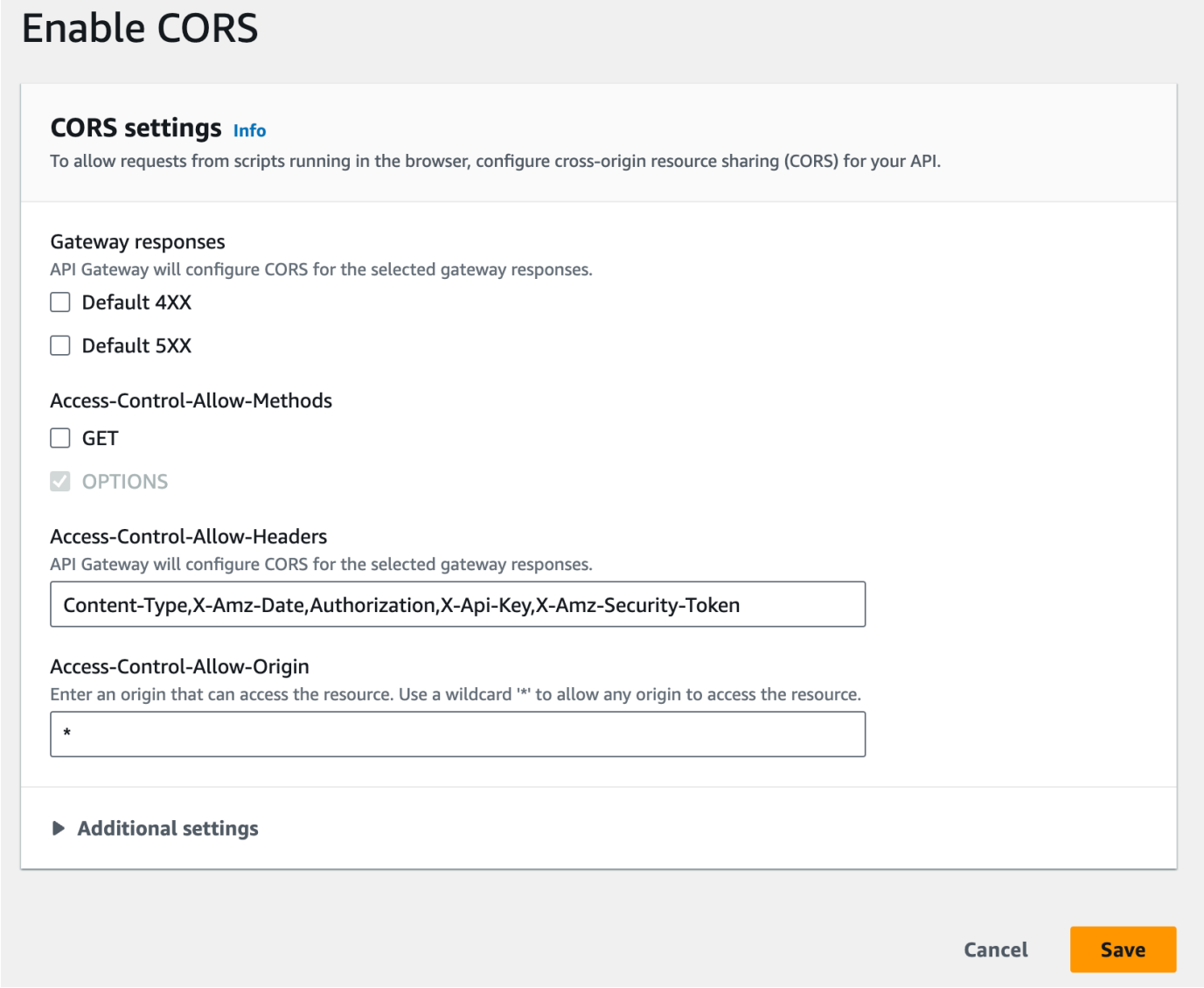
      Wenn Sie die CORS-Unterstützung für eine `ANY`-Methode aktivieren, ist CORS für alle Methoden aktiviert.

   c. Geben Sie im Eingabefeld Access-Control-Allow-Headers eine statische Zeichenfolge einer durch Kommata getrennten Liste der Header ein, die der Client in der tatsächlichen Anforderung der Ressource senden muss. Verwenden Sie die von der Konsole

bereitgestellte Header-Liste mit `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` oder geben Sie Ihren eigenen Header an.

d. Verwenden Sie den von der Konsole bereitgestellten Wert `'*'` als Access-Control-Allow-Origin-Header-Wert, um Zugriffsanforderungen von allen Ursprüngen zuzulassen, oder geben Sie Ursprünge an, die auf die Ressource zugreifen dürfen.

e. Wählen Sie Speichern.



Important

Wenn Sie die oben genannten Anweisungen auf die ANY-Methode in einer Proxy-Integration anwenden, werden keine geltenden CORS-Header festgelegt. Stattdessen

muss Ihr Backend die relevanten CORS-Header zurückgeben, z. B. `Access-Control-Allow-Origin`.

Wenn CORS für die `GET`-Methode aktiviert wurde, wird eine `OPTIONS`-Methode zur Ressource hinzugefügt, wenn sie dort noch nicht vorhanden ist. Die `200`-Antwort der `OPTIONS`-Methode wird automatisch so konfiguriert, dass sie die drei `Access-Control-Allow-*`-Header zurückgibt, um die Preflight-Handshakes zu erfüllen. Darüber hinaus ist die tatsächliche (`GET`-) Methode standardmäßig auch so konfiguriert, dass sie den `Access-Control-Allow-Origin`-Header auch in seiner 200-Antwort zurückgibt. Für andere Arten von Antworten müssen Sie sie manuell konfigurieren, sodass sie den `Access-Control-Allow-Origin'`-Header mit "*" oder bestimmten Ursprüngen zurückgeben, wenn Sie den `Cross-origin access`-Fehler nicht zurückgeben möchten.

Wenn Sie die CORS-Unterstützung für Ihre Ressource aktiviert haben, müssen Sie die API für die neuen Einstellungen (erneut) bereitstellen, damit diese wirksam werden. Weitere Informationen finden Sie unter the section called “Bereitstellen einer REST-API (Konsole)”.

> **Note**
>
> Wenn Sie die CORS-Unterstützung für Ihre Ressource nicht aktivieren können, nachdem Sie das Verfahren ausgeführt haben, empfehlen wir Ihnen, Ihre CORS-Konfiguration mit der Beispiel-API-Ressource zu vergleichen. `/pets` Informationen zum Erstellen der Beispiel-API finden Sie unter. the section called “Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels”

CORS für eine Ressource über die API Gateway-Import-API aktivieren

Wenn Sie die API Gateway-Import-API verwenden, können Sie CORS-Unterstützung mit einer OpenAPI-Datei einrichten. Sie müssen zunächst eine `OPTIONS`-Methode in Ihrer Ressource definieren, die die erforderlichen Header zurückgibt.

> **Note**
>
> Web-Browser erwarten, dass Access-Control-Allow-Header und Access-Control-Allow-Origin-Header in jeder API-Methode eingerichtet werden, die CORS-Anforderungen akzeptieren.

Darüber hinaus stellen einige Browser zuerst eine HTTP-Anforderung an eine OPTIONS-Methode in derselben Ressource und erwarten dann, dieselben Header zu erhalten.

Options-Beispielmethode

Das folgende Beispiel erstellt eine OPTIONS-Methode für eine Mock-Integration.

OpenAPI 3.0

```
/users:
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    tags:
    - CORS
    responses:
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Origin:
            schema:
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates:
        application/json: "{\"statusCode\": 200}"
      passthroughBehavior: "never"
      responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
```

```
            method.response.header.Access-Control-Allow-Origin: "'*'"
```

OpenAPI 2.0

```
/users:
   options:
      summary: CORS support
      description: |
        Enable CORS by returning correct headers
      consumes:
        - "application/json"
      produces:
        - "application/json"
      tags:
        - CORS
      x-amazon-apigateway-integration:
        type: mock
        requestTemplates: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
      responses:
        200:
          description: Default response for CORS method
          headers:
            Access-Control-Allow-Headers:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Origin:
              type: "string"
```

Nachdem Sie die `OPTIONS`-Methode für Ihre Ressource konfiguriert haben, können Sie die erforderlichen Header zu den anderen Methoden in derselben Ressource, die CORS-Anforderungen akzeptieren muss, hinzufügen.

1. Deklarieren Sie Access-Control-Allow-Origin und Headers für die Antworttypen.

   OpenAPI 3.0

   ```
   responses:
     200:
       description: Default response for CORS method
       headers:
         Access-Control-Allow-Origin:
           schema:
             type: "string"
         Access-Control-Allow-Methods:
           schema:
             type: "string"
         Access-Control-Allow-Headers:
           schema:
             type: "string"
       content: {}
   ```

   OpenAPI 2.0

   ```
   responses:
       200:
         description: Default response for CORS method
         headers:
           Access-Control-Allow-Headers:
             type: "string"
           Access-Control-Allow-Methods:
             type: "string"
           Access-Control-Allow-Origin:
             type: "string"
   ```

2. Richten Sie im `x-amazon-apigateway-integration`-Tag das Mapping für diese Header auf Ihre statischen Werte ein:

   OpenAPI 3.0

   ```
   responses:
       default:
         statusCode: "200"
         responseParameters:
   ```

```
            method.response.header.Access-Control-Allow-Headers: "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
          responseTemplates:
            application/json: |
              {}
```

OpenAPI 2.0

```
    responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
```

Beispiel-API

Im folgenden Beispiel wird eine vollständige API mit einer `OPTIONS`-Methode und einer `GET`-Methode mit einer `HTTP`-Integration erstellt.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "cors-api"
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
servers:
- url: "/{basePath}"
  variables:
    basePath:
      default: "/test"
paths:
  /:
    get:
      operationId: "GetPet"
      responses:
        "200":
```

```
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
          content: {}
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
            Access-Control-Allow-Methods:
              schema:
                type: "string"
            Access-Control-Allow-Headers:
              schema:
                type: "string"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Empty"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
```

```
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
components:
  schemas:
    Empty:
      type: "object"
```

OpenAPI 2.0

```
swagger: "2.0"
info:
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
  title: "cors-api"
basePath: "/test"
schemes:
- "https"
paths:
  /:
    get:
      operationId: "GetPet"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      consumes:
      - "application/json"
```

```
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
definitions:
  Empty:
    type: "object"
```

Testen von CORS

Sie können die CORS-Konfiguration Ihrer API testen, indem Sie Ihre API aufrufen und die CORS-Header in der Antwort überprüfen. Der folgende `curl`-Befehl sendet eine OPTIONS-Anforderung an eine bereitgestellte API.

```
curl -v -X OPTIONS https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}
```

```
< HTTP/1.1 200 OK
< Date: Tue, 19 May 2020 00:55:22 GMT
```

```
< Content-Type: application/json
< Content-Length: 0
< Connection: keep-alive
< x-amzn-RequestId: a1b2c3d4-5678-90ab-cdef-abc123
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Content-Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-
Security-Token
< x-amz-apigw-id: Abcd=
< Access-Control-Allow-Methods: DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT
```

Die Header `Access-Control-Allow-Origin`, `Access-Control-Allow-Headers` und `Access-Control-Allow-Methods` in der Antwort zeigen, dass die API CORS unterstützt. Weitere Informationen finden Sie unter Aktivieren von CORS für eine REST-API-Ressource.

Mit binären Medientypen für REST-APIs arbeiten

In API Gateway haben die API-Anfrage und -Antwort einen Text-Payload oder einen Binär-Payload. Eine Text-Nutzlast ist eine `UTF-8`-codierte JSON-Zeichenfolge. Eine binäre Nutzlast ist jede beliebige andere außer einer Text-Nutzlast. Eine binäre Nutzlast kann beispielsweise eine JPEG-, GZip- oder XML-Datei sein. Welche API-Konfiguration zur Unterstützung von binären Medien erforderlich ist, hängt davon ab, ob Ihre API Proxy- oder Nicht-Proxyintegrationen verwendet.

AWS Lambda Proxy-Integrationen

Um binäre Nutzlasten für AWS Lambda Proxy-Integrationen zu verarbeiten, müssen Sie die Antwort Ihrer Funktion base64-kodieren. Sie müssen das auch für Ihre API konfigurieren. binaryMediaTypes Die `binaryMediaTypes`-Konfiguration Ihrer API ist eine Liste von Inhaltstypen, die Ihre API als Binärdaten behandelt. Beispiele für binäre Medientypen umfassen `image/png` oder `application/octet-stream`. Sie können das Platzhalterzeichen (`*`) verwenden, um mehrere Medientypen abzudecken. `*/*` enthält beispielsweise alle Inhaltstypen.

Beispielcode finden Sie unter the section called “Rückgabe binärer Medien von einer Lambda-Proxy-Integration”.

Nicht-Proxy-Integrationen

Um binäre Payloads für Nicht-Proxy-Integrationen zu verarbeiten, fügen Sie die Medientypen der binaryMediaTypesListe der Ressource hinzu. `RestApi` Die `binaryMediaTypes`-Konfiguration Ihrer API ist eine Liste von Inhaltstypen, die Ihre API als Binärdaten behandelt. Alternativ können

Sie die ContentHandling-Eigenschaften für die Integration und die IntegrationResponseRessourcen festlegen. Der Wert `contentHandling` kann `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` oder undefiniert sein.

Je nach `contentHandling`-Wert sowie abhängig von der Tatsache, ob der `Content-Type`-Header der Antwort oder der `Accept`-Header der eingehenden Anfrage mit einem Eintrag in der Liste `binaryMediaTypes` übereinstimmt, kann Amazon API Gateway die unformatierten binären Bytes als base64-kodierte Zeichenfolge kodieren, eine base64-kodierte Zeichenfolge zurück in die Raw-Bytes dekodieren oder den Textkörper ohne Änderung weitergeben.

Um binäre Payloads für Ihre API in API Gateway zu unterstützen, müssen Sie die API wie folgt konfigurieren:

- Fügen Sie die gewünschten binären Medientypen zur `binaryMediaTypes` Liste auf der RestApiRessource hinzu. Wenn diese Eigenschaft und die `contentHandling`-Eigenschaft nicht definiert sind, werden die Nutzlasten als UTF-8-kodierte JSON-Zeichenfolgen verarbeitet.
- Legen Sie die Eigenschaft `contentHandling` der Ressource Integration fest.
  - Damit die Anforderungsnutzlast von einer Base64-kodierten Zeichenfolge in ihr binäres Blob konvertiert wird, setzen Sie die Eigenschaft auf `CONVERT_TO_BINARY`.
  - Damit die Anforderungsnutzlast von einem binären Blob in eine Base64-codierte Zeichenfolge konvertiert wird, setzen Sie die Eigenschaft auf `CONVERT_TO_TEXT`.
  - Um die Nutzlast ohne Änderung zu übergeben, lassen Sie die Eigenschaft undefiniert. Um eine binäre Nutzlast ohne Änderung zu übergeben, müssen Sie auch sicherstellen, dass der `Content-Type` mit einem der `binaryMediaTypes`-Einträge übereinstimmt und Passthrough-Verhalten für die API aktiviert sind.
- Legen Sie die `contentHandling` Eigenschaft der IntegrationResponseRessource fest. Die `contentHandling`-Eigenschaft, der `Accept`-Header in Client-Anfragen und die `binaryMediaTypes` Ihrer APIs legen zusammen fest, wie API Gateway Inhaltstypkonvertierungen handhabt. Details hierzu finden Sie unter the section called “Inhaltstypkonvertierungen in API Gateway”.

> **Important**
>
> Wenn eine Anfrage mehrere Medientypen in ihrem `Accept`-Header enthält, berücksichtigt API Gateway nur den ersten `Accept`-Medientyp. Wenn Sie die Reihenfolge der `Accept`-Medientypen nicht beeinflussen können und der Medientyp Ihres binären Inhalts nicht der

erste in der Liste ist, fügen Sie den ersten `Accept`-Medientyp in der `binaryMediaTypes`-Liste Ihrer API hinzu. API Gateway behandelt alle Inhaltstypen in dieser Liste als binär. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.

Ausführliche Informationen darüber, wie API Gateway die Text- und Binär-Payloads behandelt, finden Sie unter Inhaltstypkonvertierungen in API Gateway.

Inhaltstypkonvertierungen in API Gateway

Die Kombination aus den `binaryMediaTypes` Ihrer API, den Headern in Client-Anfragen und der Integrations-`contentHandling`-Eigenschaft bestimmt, wie API Gateway Payloads kodiert.

Die folgende Tabelle zeigt, wie API Gateway die Anforderungsnutzlast für bestimmte Konfigurationen des `Content-Type` Headers einer Anfrage, der `binaryMediaTypes` Liste einer RestApiRessource und des `contentHandling` Eigenschaftswerts der Integrationsressource konvertiert.

API-Anfragen zur Konvertierung von Inhaltstypen in API Gateway

Nutzlast der Methodenanforderung	**`Content-Type`** -Header der Anforderung	**`binaryMediaTypes`**	**`contentHandling`**	Nutzlast der Integrationsanforderung
Textdaten	Beliebiger Datentyp	Undefined	Undefined	UTF8-kodierte Zeichenfolge
Textdaten	Beliebiger Datentyp	Undefined	`CONVERT_TO_BINARY`	Base64-dekodiertes binäres Blob
Textdaten	Beliebiger Datentyp	Undefined	`CONVERT_TO_TEXT`	UTF8-kodierte Zeichenfolge
Textdaten	Ein Textdatentyp	Satz mit übereinst	Undefined	Textdaten

Nutzlast der Methodenanforderung	**Content-Type** -Header der Anforderung	**binaryMediaTypes**	**contentHandling**	Nutzlast der Integrationsanforderung
		immenden Medientypen		
Textdaten	Ein Textdatentyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Base64-dekodiertes binäres Blob
Textdaten	Ein Textdatentyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	Textdaten
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	Undefined	Binäre Daten
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Binäre Daten
Binäre Daten	Ein binärer Datentyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	Base64-kodierte Zeichenfolge

Die folgende Tabelle zeigt, wie API Gateway die Antwortnutzlast für bestimmte Konfigurationen des `Accept` Headers einer Anfrage, der `binaryMediaTypes` Liste einer RestApiRessource und des `contentHandling` Eigenschaftswerts der IntegrationResponseRessource konvertiert.

> **Important**
>
> Wenn eine Anfrage mehrere Medientypen in ihrem `Accept`-Header enthält, berücksichtigt API Gateway nur den ersten `Accept`-Medientyp. Wenn Sie die Reihenfolge der `Accept`-Medientypen nicht beeinflussen können und der Medientyp Ihres binären Inhalts nicht der erste in der Liste ist, fügen Sie den ersten `Accept`-Medientyp in der `binaryMediaTypes`-Liste Ihrer API hinzu. API Gateway behandelt alle Inhaltstypen in dieser Liste als binär. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.

API Gateway-Antwort-Inhaltstyp-Konvertierungen

Nutzlast der Integrationsantwort	**`Accept`**-Header der Anforderung	**`binaryMediaTypes`**	**`contentHandling`**	Nutzlast der Methodenantwort
Text- oder Binärdaten	Ein Texttyp	Undefined	Undefined	UTF8-kodierte Zeichenfolge
Text- oder Binärdaten	Ein Texttyp	Undefined	`CONVERT_TO_BINARY`	Base64-dekodiertes Blob
Text- oder Binärdaten	Ein Texttyp	Undefined	`CONVERT_TO_TEXT`	UTF8-kodierte Zeichenfolge
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	Undefined	Textdaten
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Base64-dekodiertes Blob

Nutzlast der Integrationsantwort	**Accept**-Header der Anforderung	**binaryMediaTypes**	**contentHandling**	Nutzlast der Methodenantwort
Textdaten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	UTF8-kodierte Zeichenfolge
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	Undefined	Base64-dekodiertes Blob
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Base64-dekodiertes Blob
Textdaten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	UTF8-kodierte Zeichenfolge
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	Undefined	Base64-kodierte Zeichenfolge
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Binäre Daten
Binäre Daten	Ein Texttyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	Base64-kodierte Zeichenfolge

Nutzlast der Integrationsantwort	**Accept**-Header der Anforderung	**binaryMediaTypes**	**contentHandling**	Nutzlast der Methodenantwort
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	Undefined	Binäre Daten
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_BINARY`	Binäre Daten
Binäre Daten	Ein Binärtyp	Satz mit übereinstimmenden Medientypen	`CONVERT_TO_TEXT`	Base64-kodierte Zeichenfolge

Bei Umwandlung einer Textnutzlast in ein binäres Blob geht API Gateway davon aus, dass es sich bei den Textdaten um eine base64-kodierte Zeichenfolge handelt, und gibt die Binärdaten als base64-dekodierten Blob zurück. Wenn die Konvertierung fehlschlägt, wird eine `500`-Antwort zurückgegeben, die auf einen API-Konfigurationsfehler hinweist. Für eine solche Konvertierung geben Sie keine Mapping-Vorlage an, auch wenn Sie das Pass-Through-Verhalten in der API aktivieren müssen.

Bei der Konvertierung eines binären Payloads in eine Textzeichenfolge wendet API Gateway immer eine base64-Kodierung auf die Binärdaten an. Sie können eine Mapping-Vorlage für eine solche Nutzlast definieren, aber nur auf die base64-kodierte Zeichenfolge in der Mapping-Vorlage durch `$input.body` zugreifen, wie im folgenden Beispielauszug einer Mapping-Vorlage dargestellt.

```
{
    "data": "$input.body"
}
```

Damit die binäre Nutzlast unverändert übergeben wird, müssen Sie das Pass-Through-Verhalten in der API aktivieren.

Binärunterstützung über die API Gateway-Konsole aktivieren

Der Abschnitt erklärt, wie die Binärunterstützung in der API Gateway-Konsole aktiviert wird. Als Beispiel verwenden wir eine API, die in Amazon S3 integriert ist. Wir konzentrieren uns darauf, die unterstützten Medientypen festzulegen und anzugeben, wie die Nutzlast verarbeitet werden soll. Ausführliche Informationen über die Erstellung einer in Amazon S3 integrierten API finden Sie unter Tutorial: Erstellen einer REST-API als Amazon S3-Proxy in API Gateway.

So aktivieren Sie Binärunterstützung mit Hilfe der API Gateway-Konsole:

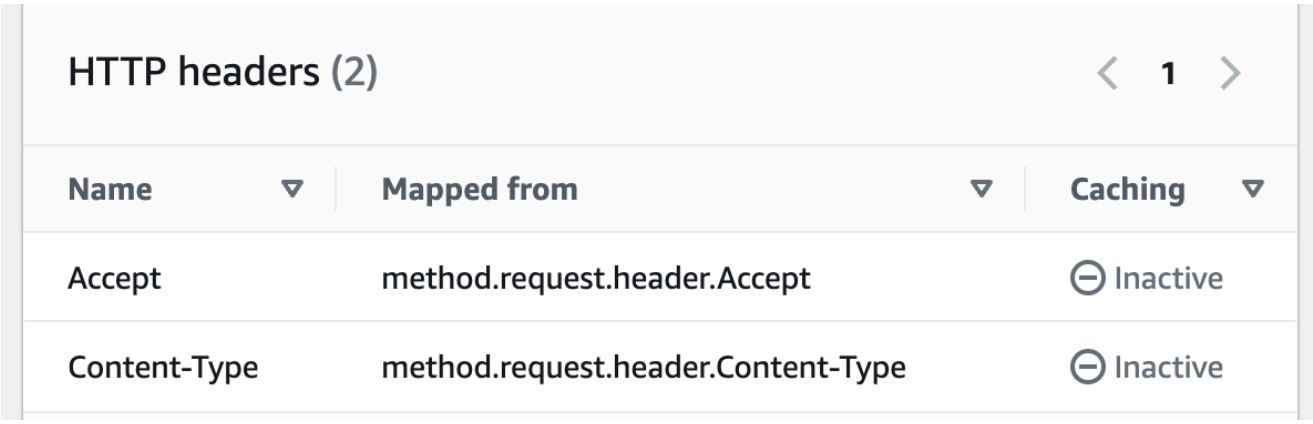
1. Festlegen der binären Medientypen für die API:
   a. Erstellen Sie eine neue API oder wählen Sie eine vorhandene API aus. In diesem Beispiel nennen wir die API `FileMan`.
   b. Klicken Sie unter der ausgewählten API im Hauptnavigationsbereich auf API-Einstellungen.
   c. Im Bereich API-Einstellungen klicken Sie dann im Abschnitt Binäre Medientypen verwalten auf Binären Medientyp hinzufügen.
   d. Klicken Sie auf Binären Medientyp hinzufügen.
   e. Geben Sie einen erforderlichen Medientyp (z. B. **`image/png`**) im Eingabetextfeld ein. Wiederholen Sie diesen Schritt bei Bedarf, um weitere Medientypen hinzuzufügen. Um alle binären Medientypen zu unterstützen, legen Sie fes `*/*`.
   f. Wählen Sie Änderungen speichern aus.
2. Legen Sie fest, wie Nachrichtennutzlasten für die API-Methode behandelt werden sollen:
   a. Erstellen Sie eine neue Ressource oder wählen Sie eine vorhandene Ressource in der API aus. In diesem Beispiel wird die Ressource `/{folder}/{item}` verwendet.
   b. Erstellen Sie eine neue Methode oder wählen Sie eine vorhandene Methode für die Ressource aus. Als Beispiel verwenden wir die `GET /{folder}/{item}`-Methode, die mit der `Object GET`-Aktion in Amazon S3 integriert ist.
   c. Wählen Sie unter Inhaltsbehandlung eine Option aus.

Wählen Sie Pass-Through, wenn der Textkörper nicht konvertiert werden soll, wenn Client und Backend das gleiche Binärformat akzeptieren. Klicken Sie auf In Text umwandeln, um den binären Text in eine base64-kodierte Zeichenfolge zu konvertieren, wenn beispielsweise eine binäre Anforderungsnutzlast als JSON-Eigenschaft für das Backend erforderlich ist. KLicken Sie auf In Binärdaten umwandeln, wenn der Client eine base64-kodierte Zeichenfolge übermittelt, aber für das Backend das ursprüngliche Binärformat erforderlich ist, oder wenn der Endpunkt eine base64-kodierte Zeichenfolge zurückgibt und der Client nur eine binäre Ausgabe akzeptiert.

d. Wählen Sie für Anforderungstext-Pass-Through die Option Wenn keine Vorlagen definiert sind (empfohlen) aus.

   Alternativ können Sie auch Nie auswählen. Dies bedeutet, dass die API Daten mit Inhaltstypen ablehnt, die keine Zuweisungsvorlage enthalten.

e. Behalten Sie den `Accept`-Header der eingehenden Anforderung in der Integrationsanforderung bei. Sie sollten dies auch tun, wenn Sie `contentHandling` auf `passthrough` festgelegt haben und diese Einstellung zur Laufzeit überschrieben werden soll.



f. Für eine Umwandlung in Text definieren Sie eine Mapping-Vorlage, um die base64-kodierten binären Daten in das erforderliche Format zu bringen.

   Im Folgenden sehen Sie ein Beispiel für eine Zuordnungsvorlage, die in Text konvertiert werden kann:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

   Das Format dieser Mapping-Vorlage hängt von den Anforderungen des Endpunkts an die Eingabe ab.

g. Wählen Sie Speichern.

Binärunterstützung über die API Gateway-REST-API aktivieren

Die folgenden Aufgaben zeigen, wie die Binärunterstützung unter Verwendung der API Gateway-REST-API-Aufrufe aktiviert werden kann.

Themen

- Hinzufügen und Aktualisieren der unterstützten binären Medientypen für eine API
- Konfigurieren der Konvertierung von Anforderungsnutzlasten
- Konfigurieren der Konvertierung von Antwortnutzlasten
- Konvertieren von Binärdaten in Textdaten
- Konvertieren von Textdaten in eine binäre Nutzlast
- Übergeben einer binären Nutzlast

Hinzufügen und Aktualisieren der unterstützten binären Medientypen für eine API

Um API Gateway die Unterstützung eines neuen binären Medientyps zu ermöglichen, müssen Sie den binären Medientyp zur `binaryMediaTypes`-Liste der `RestApi`-Ressource hinzufügen. Wenn API Gateway beispielsweise JPEG-Bilder verarbeiten soll, müssen Sie eine `PATCH`-Anfrage an die `RestApi`-Ressource stellen:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
 ]
}
```

Die MIME-Typ-Spezifikation `image/jpeg`, die Teil des `path`-Eigenschaftswerts ist, wird als `image~1jpeg` kodiert.

Um die unterstützten binären Medientypen zu aktualisieren, ersetzen oder entfernen Sie sie aus der Liste `binaryMediaTypes` der `RestApi`-Ressource. Wenn Sie beispielsweise die Binärunterstützung von JPEG-Dateien in Rohbytes ändern möchten, senden Sie wie folgt eine `PATCH`-Anforderung an die `RestApi`-Ressource:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Konfigurieren der Konvertierung von Anforderungsnutzlasten

Wenn der Endpunkt eine binäre Eingabe erfordert, legen Sie die `contentHandling`-Eigenschaft der `Integration`-Ressource auf `CONVERT_TO_BINARY` fest. Um dies zu tun, senden Sie eine `PATCH`-Anforderung wie folgt:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Konfigurieren der Konvertierung von Antwortnutzlasten

Wenn der Client das Ergebnis als binäres Blob anstelle der base64-kodierten Nutzlast, die vom Endpunkt zurückgegeben wird, akzeptiert, legen Sie die `contentHandling`-Eigenschaft für die `IntegrationResponse`-Ressource auf `CONVERT_TO_BINARY` fest. Um dies zu tun, senden Sie eine `PATCH`-Anforderung wie folgt:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Konvertieren von Binärdaten in Textdaten

Gehen Sie wie folgt vor, um Binärdaten als JSON-Eigenschaft der Eingabe an AWS Lambda oder Kinesis über API Gateway zu senden:

1. Aktivieren Sie die Unterstützung binärer Nutzlasten der API durch Hinzufügen des neuen binären Medientyps `application/octet-stream` zur Liste `binaryMediaTypes` der API.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Legen Sie `CONVERT_TO_TEXT` für die Eigenschaft `contentHandling` der `Integration`-Ressource fest und geben Sie eine Mapping-Vorlage an, um die base64-kodierte Zeichenfolge der Binärdaten einer JSON-Eigenschaft zuzuweisen. Im folgenden Beispiel ist die JSON-Eigenschaft `body` und `$input.body` enthält die base64-kodierte Zeichenfolge.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_TEXT"
    },
    {
      "op" : "add",
      "path" : "/requestTemplates/application~1octet-stream",
      "value" : "{\"body\": \"$input.body\"}"
    }
  ]
}
```

Konvertieren von Textdaten in eine binäre Nutzlast

Angenommen, eine Lambda-Funktion gibt eine Bilddatei als base64-kodierte Zeichenfolge zurück. Gehen Sie wie folgt vor, um diese Binärausgabe über API Gateway an den Client zu übergeben:

1. Aktualisieren Sie die Liste `binaryMediaTypes` der API durch Hinzufügen des binären Medientyps `application/octet-stream`, sofern dieser noch nicht in der Liste enthalten ist.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Legen Sie die `contentHandling`-Eigenschaft für die `Integration`-Ressource auf `CONVERT_TO_BINARY` fest. Definieren Sie keine Mapping-Vorlage. Wenn Sie keine Mapping-Vorlage definieren, ruft API Gateway die Passthrough-Vorlage auf, um den base64-dekodierten binären Blob als Bilddatei an den Client zurückzugeben.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Übergeben einer binären Nutzlast

Um ein Bild in einem Amazon S3-Bucket mit API Gateway zu speichern, gehen Sie wie folgt vor:

1. Aktualisieren Sie die Liste `binaryMediaTypes` der API durch Hinzufügen des binären Medientyps `application/octet-stream`, sofern dieser noch nicht in der Liste enthalten ist.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
```

```
  }
 ]
}
```

2. Legen Sie die `contentHandling`-Eigenschaft für die `Integration`-Ressource auf `CONVERT_TO_BINARY` fest. Legen Sie `WHEN_NO_MATCH` als `passthroughBehavior`-Eigenschaftswert fest, ohne eine Mapping-Vorlage zu definieren. Dadurch wird API Gateway in die Lage versetzt, die Passthrough-Vorlage aufzurufen.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    },
    {
      "op" : "replace",
      "path" : "/passthroughBehaviors",
      "value" : "WHEN_NO_MATCH"
    }
  ]
}
```

Importieren und Exportieren von Inhaltskodierungen

Um die `binaryMediaTypes` Liste in eine zu importieren RestApi, verwenden Sie die folgende API Gateway Gateway-Erweiterung zur OpenAPI-Definitionsdatei der API. Die Erweiterung wird auch verwendet, um die API-Einstellungen zu exportieren.

- x-amazon-apigateway-binaryEigenschaft -media-types

Um den `contentHandling`-Eigenschaftswert für eine `Integration`- oder `IntegrationResponse`-Ressource zu importieren und zu exportieren, verwenden Sie die folgenden API Gateway-Erweiterungen für die OpenAPI-Definitionen:

- x-amazon-apigateway-integration Objekt

- x-amazon-apigateway-integration.response-Objekt

Rückgabe binärer Medien von einer Lambda-Proxy-Integration

Zum Zurückgeben binärer Medien aus einer AWS Lambda -Proxy-Integration verwenden Sie für die Antwort Ihrer Lambda-Funktion base64-Codierung. Sie müssen auch die binären Medientypen Ihrer API konfigurieren. Die maximale Größe der Nutzlast ist 10 MB.

> **Note**
>
> Um einen Webbrowser zum Aufruf einer API mit dieser Beispielintegration zu verwenden, legen Sie die binären Medientypen Ihrer API auf fes `*/*`. API Gateway verwendet den ersten `Accept`-Header von Clients, um zu bestimmen, ob eine Antwort binäre Medien zurückgeben soll. Um binäre Medien zurückzugeben, wenn Sie die Reihenfolge der `Accept`-Headerwerte nicht steuern können, z. B. Anforderungen von einem Browser, setzen Sie die binären Medientypen der API (für alle Inhaltstypen) auf `*/*`.

Die folgende Beispiel-Lambda-Funktion kann ein binäres Bild von Amazon S3 oder Text an Clients zurückgeben. Die Antwort der Funktion enthält einen `Content-Type`-Header, der dem Client den Typ der zurückgegebenen Daten angibt. Die Funktion legt die `isBase64Encoded`-Eigenschaft bedingt in ihrer Antwort fest, abhängig von der Art der Daten, die sie zurückgibt.

Node.js

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3"

const client = new S3Client({region: 'us-east-2'});

export const handler = async (event) => {

  var randomint = function(max) {
    return Math.floor(Math.random() * max);
  }
  var number = randomint(2);
  if (number == 1){
    const input = {
      "Bucket" : "bucket-name",
      "Key" : "image.png"
      }
```

```
    try {
      const command = new GetObjectCommand(input)
      const response = await client.send(command);
      var str = await response.Body.transformToByteArray();
    } catch (err) {
      console.error(err);
    }
    const base64body = Buffer.from(str).toString('base64');
    return {
      'headers': { "Content-Type": "image/png" },
      'statusCode': 200,
      'body': base64body,
      'isBase64Encoded': true
      }
    } else {
        return {
        'headers': { "Content-Type": "text/html" },
        'statusCode': 200,
        'body': "<h1>This is text</h1>",
        }
    }
}
```

Python

```
import base64
import boto3
import json
import random

s3 = boto3.client('s3')

def lambda_handler(event, context):
    number = random.randint(0,1)
    if number == 1:
        response = s3.get_object(
            Bucket='bucket-name',
            Key='image.png',
        )
        image = response['Body'].read()
        return {
            'headers': { "Content-Type": "image/png" },
            'statusCode': 200,
```

```
            'body': base64.b64encode(image).decode('utf-8'),
            'isBase64Encoded': True
        }
    else:
        return {
            'headers': { "Content-type": "text/html" },
            'statusCode': 200,
            'body': "<h1>This is text</h1>",
        }
```

Weitere Informationen zu binären Medientypen finden Sie unter Mit binären Medientypen für REST-APIs arbeiten.

Über eine API Gateway-API auf Binärdateien in Amazon S3 zugreifen

Die folgenden Beispiele zeigen die OpenAPI-Datei, die für den Zugriff auf Bilder in Amazon S3 verwendet wird, wie man ein Bild von Amazon S3 herunterlädt und wie man ein Bild in Amazon S3 hochlädt.

Themen

- OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Amazon S3
- Bilddatei von Amazon S3 herunterladen
- Bilde in Amazon S3 hochladen

OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Amazon S3

Die folgende OpenAPI-Datei zeigt eine Beispiel-API, die das Herunterladen einer Bilddatei von Amazon S3 und das Hochladen einer Bilddatei in Amazon S3 illustriert. Diese API stellt die Methoden `GET /s3?key={file-name}` und `PUT /s3?key={file-name}` zum Herunter- und Hochladen einer bestimmten Bilddatei bereit. Die Methode `GET` gibt die Bilddatei als base64-kodierte Zeichenfolge als Teil einer JSON-Ausgabe zurück, und zwar nach der bereitgestellten Mapping-Vorlage, in einer Antwort "200 OK". Die Methode `PUT` übernimmt einen unformatierten binären Blob als Eingabe und gibt eine Antwort "200 OK" mit einer leeren Nutzlast zurück.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
```

```
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/s3": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.key": "method.request.querystring.key"
               },
```

```
            "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
            "passthroughBehavior": "when_no_match",
            "httpMethod": "GET",
            "type": "aws"
         }
      },
      "put": {
         "parameters": [
            {
               "name": "key",
               "in": "query",
               "required": false,
               "schema": {
                  "type": "string"
               }
            }
         ],
         "responses": {
            "200": {
               "description": "200 response",
               "content": {
                  "application/json": {
                     "schema": {
                        "$ref": "#/components/schemas/Empty"
                     }
                  },
                  "application/octet-stream": {
                     "schema": {
                        "$ref": "#/components/schemas/Empty"
                     }
                  }
               }
            },
            "500": {
               "description": "500 response"
            }
         },
         "x-amazon-apigateway-integration": {
            "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
            "responses": {
               "default": {
                  "statusCode": "500"
               },
               "2\\d{2}": {
```

```
                    "statusCode": "200"
                }
            },
            "requestParameters": {
                "integration.request.path.key": "method.request.querystring.key"
            },
            "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
            "passthroughBehavior": "when_no_match",
            "httpMethod": "PUT",
            "type": "aws",
            "contentHandling": "CONVERT_TO_BINARY"
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types": [
    "application/octet-stream",
    "image/jpeg"
  ],
  "servers": [
    {
      "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
      "variables": {
        "basePath": {
          "default": "/v1"
        }
      }
    }
  ],
  "components": {
    "schemas": {
      "Empty": {
        "type": "object",
        "title": "Empty Schema"
      }
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
```

```
"info": {
  "version": "2016-10-21T17:26:28Z",
  "title": "ApiName"
},
"host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
"basePath": "/v1",
"schemes": [
  "https"
],
"paths": {
  "/s3": {
    "get": {
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "key",
          "in": "query",
          "required": false,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          }
        },
        "500": {
          "description": "500 response"
        }
      },
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
        "responses": {
          "default": {
            "statusCode": "500"
          },
          "2\\d{2}": {
            "statusCode": "200"           }
        },
        "requestParameters": {
```

```
            "integration.request.path.key": "method.request.querystring.key"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "aws"
        }
      },
      "put": {
        "produces": [
          "application/json", "application/octet-stream"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.key": "method.request.querystring.key"
          },
```

```
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "PUT",
          "type": "aws",
          "contentHandling" : "CONVERT_TO_BINARY"
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Bilddatei von Amazon S3 herunterladen

So laden Sie eine Bilddatei (`image.jpg`) als binären Blob von Amazon S3 herunter:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Die erfolgreiche Antwort schaut wie folgt aus:

```
200 OK HTTP/1.1

[raw bytes]
```

Die Rohbytes werden zurückgegeben, da der `Accept`-Header auf den binären Medientyp `application/octet-stream` und die Binärunterstützung für die API aktiviert ist.

Eine Alternative besteht darin, die Bilddatei (`image.jpg`) als eine base64-kodierte Zeichenfolge (formatiert als JSON-Eigenschaft) von Amazon S3 herunterzuladen. Fügen Sie der 200-Integrationsantwort eine Antwortvorlage hinzu, wie im folgenden fettgedruckten OpenAPI-Definitionsblock gezeigt:

```
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "{\n   \"image\": \"$input.body\"\n}"
              }
            }
          },
```

Die Anforderung zum Herunterladen der Bilddatei sieht folgendermaßen aus:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Bilde in Amazon S3 hochladen

So laden Sie eine Bilddatei (`image.jpg`) als binären Blob in Amazon S3 hoch:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge in Amazon S3 hoch:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Die Eingabenutzlast muss eine base64-kodierte Zeichenfolge sein, da der `Content-Type`-Headerwert auf `application/json` festgelegt ist. Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1
```

Über eine API Gateway-API auf Binärdateien in Lambda zugreifen

Das folgende OpenAPI-Beispiel zeigt, wie Sie AWS Lambda über eine API-Gateway-API auf eine Binärdatei zugreifen. Diese API stellt die `PUT /lambda?key={file-name}` Methoden zum Herunterladen `GET /lambda?key={file-name}` und Hochladen einer bestimmten Bilddatei bereit. Die Methode `GET` gibt die Bilddatei als base64-kodierte Zeichenfolge als Teil einer JSON-Ausgabe zurück, und zwar nach der bereitgestellten Mapping-Vorlage, in einer Antwort "200 OK". Die Methode `PUT` übernimmt einen unformatierten binären Blob als Eingabe und gibt eine Antwort "200 OK" mit einer leeren Nutzlast zurück.

Sie erstellen die Lambda-Funktion, die Ihre API aufruft, und sie muss eine Base64-kodierte Zeichenfolge mit dem Header von zurückgeben. `Content-Type application/json`

Themen

- OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Lambda
- Bilddatei von Lambda herunterladen
- Bild in Lambda hochladen

OpenAPI-Datei einer Beispiel-API für den Zugriff auf Bilder in Lambda

Die folgende OpenAPI-Datei zeigt eine Beispiel-API, die das Herunterladen einer Bilddatei von Lambda und das Hochladen einer Bilddatei in Lambda veranschaulicht.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/lambda": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
```

```
            "type": "AWS",
            "credentials": "arn:aws:iam::123456789012:role/Lambda",
            "httpMethod": "POST",
            "requestTemplates": {
               "application/json": "{\n   \"imageKey\":
\"$input.params('key')\"\n}"
            },
            "responses": {
               "default": {
                  "statusCode": "500"
               },
               "2\\d{2}": {
                  "statusCode": "200",
                  "responseTemplates": {
                     "application/json": "{\n   \"image\": \"$input.body\"\n}"
                  }
               }
            }
         }
      },
      "put": {
         "parameters": [
            {
               "name": "key",
               "in": "query",
               "required": false,
               "schema": {
                  "type": "string"
               }
            }
         ],
         "responses": {
            "200": {
               "description": "200 response",
               "content": {
                  "application/json": {
                     "schema": {
                        "$ref": "#/components/schemas/Empty"
                     }
                  },
                  "application/octet-stream": {
                     "schema": {
                        "$ref": "#/components/schemas/Empty"
                     }
```

```
                    }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
               "type": "AWS",
               "credentials": "arn:aws:iam::123456789012:role/Lambda",
               "httpMethod": "POST",
               "contentHandling": "CONVERT_TO_TEXT",
               "requestTemplates": {
                  "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
               },
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               }
            }
         }
      }
   },
   "x-amazon-apigateway-binary-media-types": [
      "application/octet-stream",
      "image/jpeg"
   ],
   "servers": [
      {
         "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/v1"
            }
         }
      }
   ],
```

```
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
```

```
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "{\n   \"image\": \"$input.body\"\n}"
              }
            }
          }
        }
      },
      "put": {
        "produces": [
          "application/json", "application/octet-stream"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
```

```
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "contentHandling" : "CONVERT_TO_TEXT",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Bilddatei von Lambda herunterladen

So laden Sie eine Bilddatei (`image.jpg`) als binäres Blob von Lambda herunter:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

[raw bytes]
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge, formatiert als JSON-Eigenschaft, von Lambda herunter:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Bild in Lambda hochladen

So laden Sie eine Bilddatei (`image.jpg`) als binäres Blob in Lambda hoch:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK
```

So laden Sie eine Bilddatei (`image.jpg`) als base64-kodierte Zeichenfolge in Lambda hoch:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Die erfolgreiche Antwort sieht folgendermaßen aus:

```
200 OK
```

REST-API im Amazon API Gateway aufrufen

Um eine bereitgestellte API aufzurufen, senden Kunden Anfragen an die URL für den API Gateway-Komponenten-Service zur API-Ausführung (bekannt als `execute-api`).

Die Basis-URL für REST-API liegt im folgenden Format vor:

```
https://restapi_id.execute-api.region.amazonaws.com/stage_name/
```

wobei *`restapi_id` die API-ID*, *`region` die Region* und *`stage_name` der AWS Phasenname der API-Bereitstellung* ist.

> **Important**
>
> Bevor Sie eine API aufrufen können, müssen Sie sie in API Gateway bereitstellen. Anweisungen zur Bereitstellung einer API finden Sie unter. Bereitstellen einer REST-API in Amazon API Gateway

Themen

- Abrufen der Aufruf-URL einer API
- Aufrufen einer API

- API Gateway-Konsole zum Testen einer REST-API-Methode verwenden
- Von API Gateway generiertes Java-SDK für eine REST-API verwenden
- Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden
- Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API
- Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden
- Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden

Abrufen der Aufruf-URL einer API

Sie können die Konsole AWS CLI, die oder eine exportierte OpenAPI-Definition verwenden, um die Aufruf-URL einer API abzurufen.

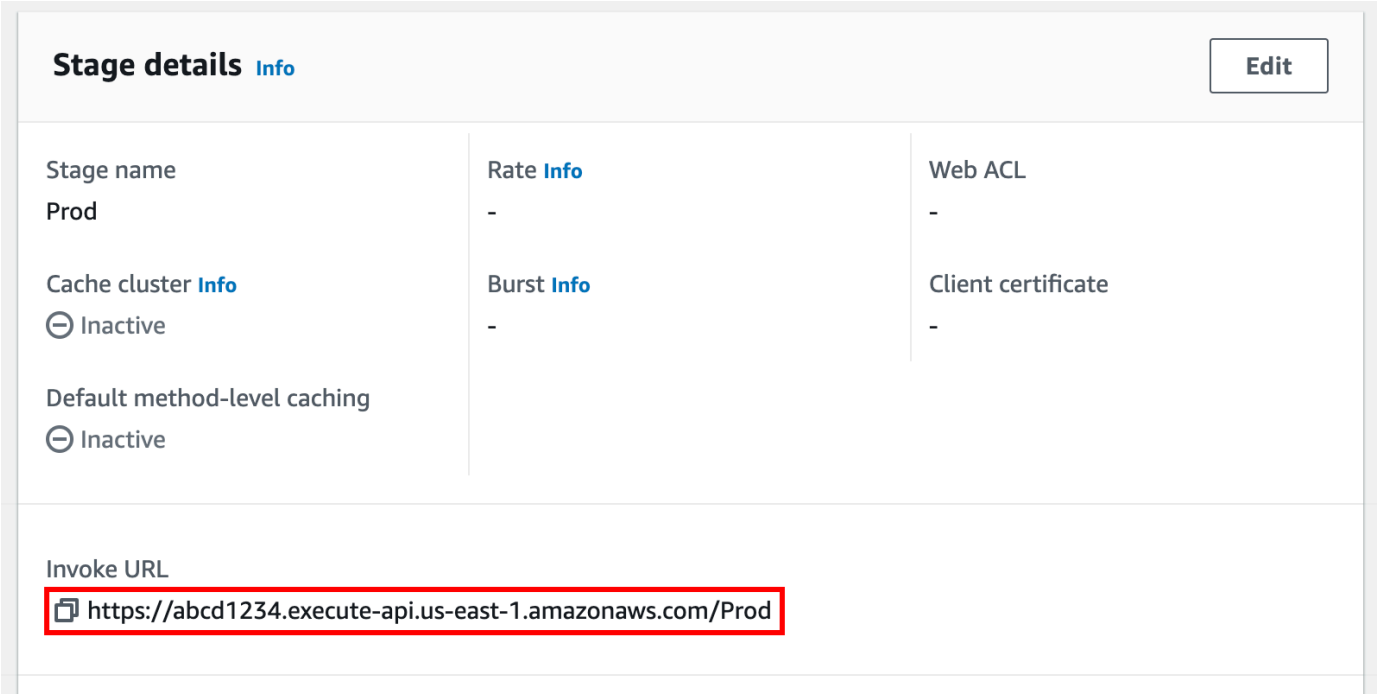
Abrufen der Aufruf-URL einer API mithilfe der Konsole

Das folgende Verfahren zeigt, wie Sie die Aufruf-URL einer API in der REST-API-Konsole abrufen.

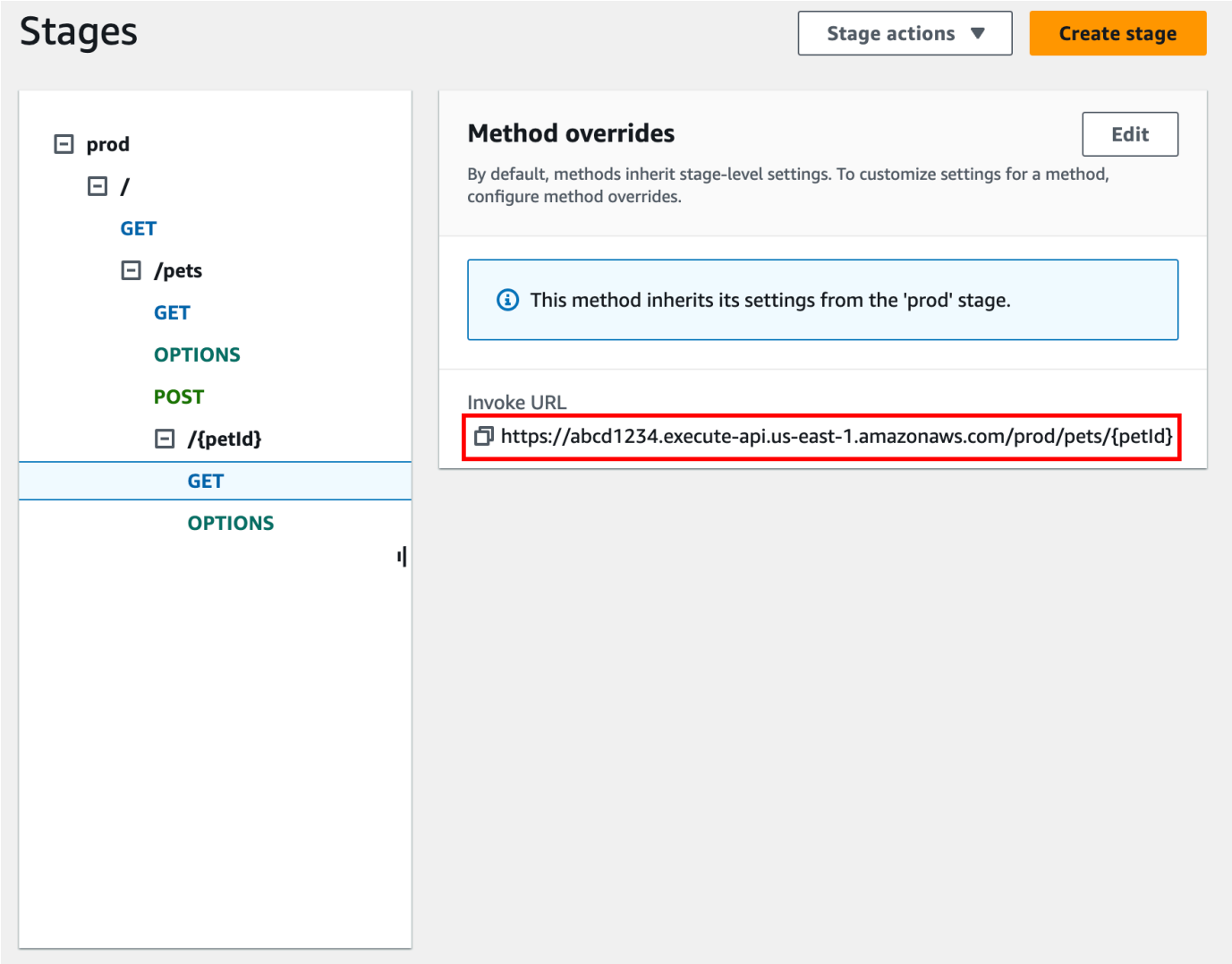
So rufen Sie die Aufruf-URL einer API mithilfe der REST-API-Konsole ab

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine bereitgestellte API aus.
3. Wählen Sie im Haupt-Navigationsbereich Stufe aus.
4. Wählen Sie unter Stufendetails das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren.

   Diese URL ist für die Root-Ressource Ihrer API.



5. Um die Aufruf-URL einer API für eine andere Ressource in Ihrer API abzurufen, erweitern Sie die Stufe unter dem sekundären Navigationsbereich und wählen Sie dann eine Methode aus.
6. Wählen Sie das Symbol „Kopieren“, um die Aufruf-URL Ihrer API auf Ressourcenebene zu kopieren.



Abrufen der Aufruf-URL einer API mithilfe von AWS CLI

Das folgende Verfahren zeigt, wie Sie die Aufruf-URL einer API mithilfe von abrufen. AWS CLI

Um die Aufruf-URL einer API mit dem abzurufen AWS CLI

1. Verwenden Sie den folgenden Befehl, um die zu erhalten. `rest-api-id` Dieser Befehl gibt alle `rest-api-id` Werte in Ihrer Region zurück. Weitere Informationen finden Sie unter get-rest-apis.

```
aws apigateway get-rest-apis
```

2. Ersetze das Beispiel `rest-api-id` durch dein`rest-api-id`, ersetze das Beispiel *`{stage-name}`* durch deinen *`{stage-name}`* und ersetze *`{region} durch deine Region`*.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Abrufen der Aufruf-URL einer API mithilfe der exportierten OpenAPI-Definitionsdatei der API

Sie können die Stamm-URL auch erstellen, indem Sie die `basePath` Felder `host` und einer exportierten OpenAPI-Definitionsdatei der API kombinieren. Anweisungen zum Exportieren Ihrer API finden Sie unterthe section called “Exportieren einer REST-API”.

Aufrufen einer API

Sie können Ihre bereitgestellte API über den Browser, Curl oder andere Anwendungen wie Postman aufrufen.

Darüber hinaus können Sie die API Gateway Gateway-Konsole verwenden, um einen API-Aufruf zu testen. Test verwendet die `TestInvoke` Funktion des API-Gateways, die API-Tests vor der Bereitstellung der API ermöglicht. Weitere Informationen finden Sie unter the section called “Testen einer REST-API-Methode mithilfe der Konsole”.

> **Note**
>
> In Parameterwerten für Abfragezeichenfolgen in einer Aufruf-URL ist `%%` nicht zulässig.

Aufrufen einer API mit einem Webbrowser

Wenn Ihre API anonymen Zugriff zulässt, können Sie einen beliebigen Webbrowser verwenden, um eine beliebige `GET` Methode aufzurufen. Geben Sie die vollständige Aufruf-URL in die Adressleiste des Browsers ein.

Für andere Methoden oder Aufrufe, für die eine Authentifizierung erforderlich ist, müssen Sie eine Nutzlast angeben oder die Anfragen signieren. Sie können diese in einem Skript hinter einer HTML-Seite oder in einer Client-Anwendung mithilfe eines der SDKs verarbeiten. AWS

Eine API mit curl aufrufen

Sie können ein Tool wie curl in Ihrem Terminal verwenden, um Ihre API aufzurufen. Der folgende CURL-Beispielbefehl ruft die GET-Methode für die `getUsers` Ressource der `prod` Stufe einer API auf.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers"
```

API Gateway-Konsole zum Testen einer REST-API-Methode verwenden

API Gateway-Konsole zum Testen einer REST-API-Methode verwenden

Themen

- Voraussetzungen
- Methode mit der API Gateway-Konsole testen

Voraussetzungen

- Sie müssen die Einstellungen für die Methoden angeben, die Sie testen möchten. Folgen Sie den Anweisungen in Methoden für REST-APIs in API Gateway.

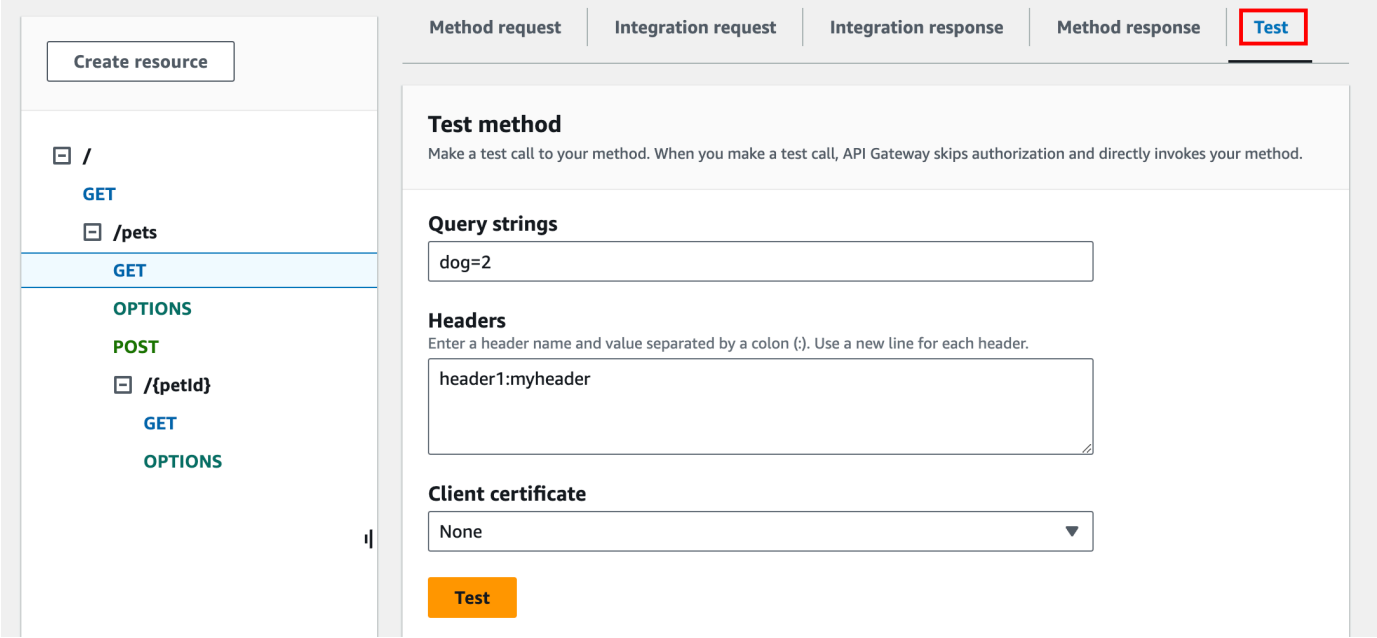
Methode mit der API Gateway-Konsole testen

> **Important**
>
> Das Testen von Methoden mit der API Gateway-Konsole kann zu Änderungen an Ressourcen führen, die nicht rückgängig gemacht werden können. Das Testen einer Methode mit der API Gateway-Konsole ist dasselbe wie der Aufruf der Methode außerhalb der API Gateway-Konsole. Wenn Sie beispielsweise über die Amazon API Gateway-Konsole eine Methode aufrufen, die die Ressourcen einer API löscht, werden diese API-Ressourcen bei einem erfolgreichem Methodenaufruf gelöscht.

So testen Sie eine Methode

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Bereich Resources auf die Methode, die Sie testen möchten.
4. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.



Geben Sie Werte in eines der angezeigten Felder ein (z. B. Abfragezeichenfolgen, Header und Anforderungstext). Die Konsole schließt diese Werte im Standard-Anwendungs-/JSON-Format in die Methodenanforderung ein.

Wenn Sie weitere Optionen angeben müssen, wenden Sie sich an den API-Eigentümer.

5. Wählen Sie Test aus. Sie bekommen die folgenden Informationen angezeigt:

   - Request ist der Pfad der Ressource, die für die Methode aufgerufen wurde.
   - Status ist der HTTP-Statuscode der Antwort.
   - Die Latenz (ms) ist die Zeit zwischen dem Eingang der Anfrage vom Anrufer und der zurückgegebenen Antwort.
   - Antworttext ist der Text der HTTP-Antwort.
   - Antwort-Header sind die Header der HTTP-Antwort.

> **Tip**
>
> Je nach Zuordnung können sich der HTTP-Statuscode, der Antworttext und die Antwortheader von denen unterscheiden, die von der Lambda-Funktion, dem HTTP-Proxy oder dem AWS Dienstproxy gesendet wurden.

- Logs sind die simulierten Amazon CloudWatch Logs-Einträge, die geschrieben worden wären, wenn diese Methode außerhalb der API Gateway Gateway-Konsole aufgerufen worden wäre.

> **Note**
>
> Obwohl die CloudWatch Logs-Einträge simuliert sind, sind die Ergebnisse des Methodenaufrufs real.

Neben der API Gateway-Konsole können Sie auch ein AWS SDK für API Gateway verwenden AWS CLI , um das Aufrufen einer Methode zu testen. Informationen dazu mithilfe von finden Sie AWS CLI unter test-invoke-method.

Von API Gateway generiertes Java-SDK für eine REST-API verwenden

In diesem Abschnitt werden die Schritte zur Verwendung eines von API Gateway generierten Java-SDKs für eine REST-API unter Verwendung der Simple Calculator-API als Beispiel vorgestellt. Bevor Sie fortfahren, müssen Sie die in Generieren eines SDK für eine REST-API in API Gateway beschriebenen Schritte durchführen.

So installieren und verwenden Sie ein von API Gateway generiertes Java-SDK:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.
2. Laden Sie Apache Maven (Version 3.5 oder höher) herunter, und installieren Sie es.
3. Laden Sie JDK 8 herunter und installieren Sie die Software.
4. Legen Sie die `JAVA_HOME`-Umgebungsvariable fest.
5. Öffnen Sie den entpackten SDK-Ordner, in dem die Datei pom.xml gespeichert ist. Dieser Ordner heißt standardmäßig `generated-code`. Führen Sie den Befehl mvn install aus, um die kompilierten Artefaktdateien in Ihrem lokalen Maven-Repository zu installieren. Dadurch wird ein Ordner namens `target` erzeugt, der die kompilierte SDK-Bibliothek enthält.

6. Geben Sie den folgenden Befehl in einem leeren Verzeichnis ein, um ein Client-Projekt-Stub zu erstellen, um die API über die installierte SDK-Bibliothek aufzurufen.

```
mvn -B archetype:generate \
    -DarchetypeGroupdId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

> **Note**
>
> Das Trennzeichen \ in dem vorausgehenden Befehl dient der besseren Lesbarkeit. Der gesamte Befehl sollte in einer einzelnen Zeile ohne das Trennzeichen stehen.

Mit diesem Befehl wird ein Anwendungs-Stub erstellt. Der Anwendungsstub enthält eine `pom.xml` Datei und einen `src` Ordner im Stammverzeichnis des Projekts (*`SimpleCalc-sdkClient`* im vorherigen Befehl). Anfänglich gibt es zwei Quelldateien: `src/main/java/`*`{package-path}`*`/App.java` und `src/test/java/`*`{package-path}`*`/AppTest.java`. In diesem Beispiel ist *`{package-path}`* `examples/aws/apig/simpleCalc/sdk/app`. Dieser Paketpfad ist von dem Wert `DarchetypeGroupdId` abgeleitet. Sie können die Datei `App.java` als Vorlage für Ihre Client-Anwendung verwenden, und Sie können nach Bedarf weitere in demselben Ordner hinzufügen. Sie können die Datei `AppTest.java` als Testvorlage für Ihre Anwendung benutzen, und Sie können nach Bedarf weitere Testcode-Dateien in demselben Ordner hinzufügen.

7. Aktualisieren Sie die Paket-Abhängigkeiten in der generierten Datei `pom.xml` wie folgt; ersetzen Sie dabei nötigenfalls die Eigenschaften `groupId`, `artifactId`, `version` und `name` Ihres Projekts:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/
POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>examples.aws.apig.simpleCalc.sdk.app</groupId>
  <artifactId>SimpleCalc-sdkClient</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>SimpleCalc-sdkClient</name>
  <url>http://maven.apache.org</url>
```

```
   <dependencies>
      <dependency>
          <groupId>com.amazonaws</groupId>
          <artifactId>aws-java-sdk-core</artifactId>
          <version>1.11.94</version>
      </dependency>
      <dependency>
          <groupId>my-apig-api-examples</groupId>
          <artifactId>simple-calc-sdk</artifactId>
          <version>1.0.0</version>
      </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.5</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.5.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Wenn eine neuere Version des abhängigen Artefakts von `aws-java-sdk-core` nicht mit der Version des oben angegebenen (`1.11.94`) kompatibel ist, müssen Sie das `<version>`-Tag auf die neue Version aktualisieren.

8. Als Nächstes zeigen wir, wie Sie die API mithilfe des SDK aufrufen, indem Sie die SDK-Methoden `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` und `postApiRoot(PostApiRootRequest req)` aufrufen. Diese Methoden entsprechen den Methoden `GET /{a}/{b}/{op}`, `GET /?a={x}&b={y}&op={operator}` und `POST /`, mit einer jeweiligen Nutzlast von `{"a": x, "b": y, "op": "operator"}`-API-Anforderungen.

   Aktualisieren Sie die Datei `App.java` wie folgt:

```
package examples.aws.apig.simpleCalc.sdk.app;

import java.io.IOException;

import com.amazonaws.opensdk.config.ConnectionConfiguration;
import com.amazonaws.opensdk.config.TimeoutConfiguration;

import examples.aws.apig.simpleCalc.sdk.*;
import examples.aws.apig.simpleCalc.sdk.model.*;
import examples.aws.apig.simpleCalc.sdk.SimpleCalcSdk.*;

public class App
{
    SimpleCalcSdk sdkClient;

    public App() {
        initSdk();
    }

    // The configuration settings are for illustration purposes and may not be a
 recommended best practice.
    private void initSdk() {
        sdkClient = SimpleCalcSdk.builder()
              .connectionConfiguration(
                  new ConnectionConfiguration()
                        .maxConnections(100)
                        .connectionMaxIdleMillis(1000))
```

```
              .timeoutConfiguration(
                  new TimeoutConfiguration()
                        .httpRequestTimeout(3000)
                        .totalExecutionTimeout(10000)
                        .socketTimeout(2000))
        .build();

   }
   // Calling shutdown is not necessary unless you want to exert explicit control
of this resource.
   public void shutdown() {
       sdkClient.shutdown();
   }

   // GetABOpResult getABOp(GetABOpRequest getABOpRequest)
   public Output getResultWithPathParameters(String x, String y, String operator)
{
    operator = operator.equals("+") ? "add" : operator;
    operator = operator.equals("/") ? "div" : operator;

       GetABOpResult abopResult = sdkClient.getABOp(new
GetABOpRequest().a(x).b(y).op(operator));
       return abopResult.getResult().getOutput();
   }

   public Output getResultWithQueryParameters(String a, String b, String op) {
       GetApiRootResult rootResult = sdkClient.getApiRoot(new
GetApiRootRequest().a(a).b(b).op(op));
       return rootResult.getResult().getOutput();
   }

   public Output getResultByPostInputBody(Double x, Double y, String o) {
    PostApiRootResult postResult = sdkClient.postApiRoot(
     new PostApiRootRequest().input(new Input().a(x).b(y).op(o)));
    return postResult.getResult().getOutput();
   }

   public static void main( String[] args )
   {
       System.out.println( "Simple calc" );
       // to begin
       App calc = new App();

       // call the SimpleCalc API
```

```
        Output res = calc.getResultWithPathParameters("1", "2", "-");
        System.out.printf("GET /1/2/-: %s\n", res.getC());

        // Use the type query parameter
        res = calc.getResultWithQueryParameters("1", "2", "+");
        System.out.printf("GET /?a=1&b=2&op=+: %s\n", res.getC());

        // Call POST with an Input body.
        res = calc.getResultByPostInputBody(1.0, 2.0, "*");
        System.out.printf("PUT /\n\n{\"a\":1, \"b\":2,\"op\":\"*\"}\n %s\n",
 res.getC());


    }
}
```

Im vorherigen Beispiel dienen die zum Instanziieren des SDK-Clients verwendeten Konfigurationseinstellungen zur Veranschaulichung und sind nicht notwendigerweise als empfohlene bewährte Methode zu verstehen. Darüber hinaus erfolgt das Aufrufen von `sdkClient.shutdown()` optional, besonders dann, wenn Sie eine genaue Kontrolle über das Freisetzen von Ressourcen benötigen.

Wir haben die wichtigsten Muster zum Aufrufen einer API über ein Java SDK gezeigt. Sie können die Anweisungen auch zum Aufrufen anderer API-Methoden verwenden.

Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden

In diesem Abschnitt beschreiben wir die Schritte zur Verwendung eines Android-SDKs, das mit API Gateway für eine REST-API generiert wurde. Bevor Sie fortfahren, müssen Sie die unter Generieren eines SDK für eine REST-API in API Gateway beschriebenen Schritte ausführen.

> **Note**
>
> Das generierte SDK ist nicht mit Android 4.4 und früheren Versionen kompatibel. Weitere Informationen finden Sie unter the section called “Wichtige Hinweise”.

So installieren und verwenden Sie ein Android-SDK, das von API Gateway generiert wurde:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.
2. Laden Sie Apache Maven (vorzugsweise Version 3.x) herunter und installieren Sie das Programm.
3. Laden Sie JDK 8 herunter und installieren Sie die Software.
4. Legen Sie die `JAVA_HOME`-Umgebungsvariable fest.
5. Führen Sie den Befehl mvn install aus, um die kompilierten Artefaktdateien in Ihrem lokalen Maven-Repository zu installieren. Dadurch wird ein Ordner namens `target` erzeugt, der die kompilierte SDK-Bibliothek enthält.
6. Kopieren Sie die SDK-Datei (ihr Name leitet sich von der Artifact-ID und der Artifact-Version ab, die Sie beim Generieren des SDK angegeben haben, z. B. `simple-calcsdk-1.0.0.jar`) aus dem Ordner `target` sowie alle anderen Bibliotheken aus dem Ordner `target/lib` in den Ordner `lib` in Ihrem Projekt.

   Bei Verwendung von Android Studio erstellen Sie unterhalb des Client-App-Moduls einen Ordner namens `libs` und kopieren Sie die erforderliche JAR-Datei in diesen Ordner. Überprüfen Sie, ob der Abschnitt mit den Abhängigkeiten in der gradle-Datei folgende Informationen enthält.

   ```
       compile fileTree(include: ['*.jar'], dir: 'libs')
       compile fileTree(include: ['*.jar'], dir: 'app/libs')
   ```

   Stellen Sie sicher, dass keine doppelten JAR-Dateien deklariert werden.
7. Verwenden Sie die `ApiClientFactory`-Klasse, um das von API Gateway generierte SDK zu initialisieren. Beispielsweise:

   ```
   ApiClientFactory factory = new ApiClientFactory();

   // Create an instance of your SDK. Here, 'SimpleCalcClient.java' is the compiled
    java class for the SDK generated by API Gateway.
   final SimpleCalcClient client = factory.build(SimpleCalcClient.class);

   // Invoke a method:
   //   For the 'GET /?a=1&b=2&op=+' method exposed by the API, you can invoke it by
    calling the following SDK method:

   Result output = client.rootGet("1", "2", "+");
   ```

```
//     where the Result class of the SDK corresponds to the Result model of the
 API.
//

//   For the 'GET /{a}/{b}/{op}'  method exposed by the API, you can call the
 following SDK method to invoke the request,

Result output = client.aBOpGet(a, b, c);

//     where a, b, c can be "1", "2", "add", respectively.

//   For the following API method:
//        POST /
//        host: ...
//        Content-Type: application/json
//
//        { "a": 1, "b": 2, "op": "+" }
// you can call invoke it by calling the rootPost method of the SDK as follows:
Input body = new Input();
input.a=1;
input.b=2;
input.op="+";
Result output = client.rootPost(body);

//      where the Input class of the SDK corresponds to the Input model of the API.

// Parse the result:
//     If the 'Result' object is { "a": 1, "b": 2, "op": "add", "c":3"}, you
 retrieve the result 'c') as

String result=output.c;
```

8. Um einen Amazon Cognito Cognito-Anmeldeinformationsanbieter zu verwenden, um Aufrufe an Ihre API zu autorisieren, verwenden Sie die `ApiClientFactory` Klasse, um mithilfe des von API Gateway generierten SDK eine Reihe von AWS Anmeldeinformationen zu übergeben, wie im folgenden Beispiel gezeigt.

```
// Use CognitoCachingCredentialsProvider to provide AWS credentials
// for the ApiClientFactory
AWSCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
```

```
        context,          // activity context
        "identityPoolId", // Cognito identity pool id
        Regions.US_EAST_1 // region of Cognito identity pool
);

ApiClientFactory factory = new ApiClientFactory()
  .credentialsProvider(credentialsProvider);
```

9. Zum Festlegen eines API-Schlüssels unter Verwendung des von API Gateway generierten SDKs verwenden Sie einen Code ähnlich dem folgenden.

```
ApiClientFactory factory = new ApiClientFactory()
  .apiKey("YOUR_API_KEY");
```

Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API

> **Note**
>
> Für diese Anleitungen müssen Sie die Anleitungen unter Generieren eines SDK für eine REST-API in API Gateway ausgeführt haben.

> **Important**
>
> Wenn für Ihre API nur ANY-Methoden definiert wurden, enthält das generierte SDK-Paket keine `apigClient.js`-Datei und Sie müssen die ANY-Methoden selbst definieren.

Um ein von API Gateway generiertes JavaScript SDK für eine REST-API zu installieren, zu initiieren und aufzurufen

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben.

2. Aktivieren Sie Cross-Origin Resource Sharing (CORS) für alle Methoden, die das von API Gateway generierte SDK aufrufen werden. Anweisungen finden Sie unter Aktivieren von CORS für eine REST-API-Ressource.

3. Bauen Sie in Ihre Webseite Verweise auf die folgenden Scripts ein.

```
<script type="text/javascript" src="lib/axios/dist/axios.standalone.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/hmac-sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/hmac.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/enc-base64.js"></
script>
<script type="text/javascript" src="lib/url-template/url-template.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/sigV4Client.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/apiGatewayClient.js"></
script>
<script type="text/javascript" src="lib/apiGatewayCore/simpleHttpClient.js"></
script>
<script type="text/javascript" src="lib/apiGatewayCore/utils.js"></script>
<script type="text/javascript" src="apigClient.js"></script>
```

4. Initialisieren Sie in Ihrem Code das von API Gateway generierte SDK, indem Sie Code ähnlich dem folgenden verwenden.

```
var apigClient = apigClientFactory.newClient();
```

Verwenden Sie Code, der dem folgenden ähnelt, um das von API Gateway generierte SDK mit AWS Anmeldeinformationen zu initialisieren. Wenn Sie AWS Anmeldeinformationen verwenden, werden alle Anfragen an die API signiert.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Um einen API-Schlüssel mit dem von API Gateway generierten SDK zu verwenden, übergeben Sie den API-Schlüssel als Parameter an das `Factory`-Objekt, indem Sie Code ähnlich dem folgenden verwenden. Wenn Sie einen API-Schlüssel verwenden, ist dieser als Teil des `x-api-key`-Headers spezifiziert, und alle Anforderungen an die API werden signiert. Das bedeutet, dass Sie die entsprechenden CORS Accept-Header für jede Anforderung festlegen müssen.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Rufen Sie die API-Methoden in API Gateway auf, indem Sie Code ähnlich dem folgenden verwenden. Jeder Aufruf gibt eine Zusage mit erfolgreichen und fehlgeschlagenen Callbacks zurück.

```
var params = {
  // This is where any modeled request parameters should be added.
  // The key is the parameter name, as it is defined in the API in API Gateway.
  param0: '',
  param1: ''
};

var body = {
  // This is where you define the body of the request,
};

var additionalParams = {
  // If there are any unmodeled query parameters or headers that must be
  //   sent with the request, add them here.
  headers: {
    param0: '',
    param1: ''
  },
  queryParams: {
    param0: '',
    param1: ''
  }
};

apigClient.methodName(params, body, additionalParams)
    .then(function(result){
      // Add success callback code here.
    }).catch( function(result){
      // Add error callback code here.
    });
```

Hier wird *methodName* aus dem Ressourcenpfad der Methodenanforderung und dem HTTP-Verb gebildet. Für die SimpleCalc API die SDK-Methoden für die API-Methoden von

```
1. GET /?a=...&b=...&op=...
2. POST /

   { "a": ..., "b": ..., "op": ...}
3. GET /{a}/{b}/{op}
```

Die entsprechenden SDK-Methoden sind:

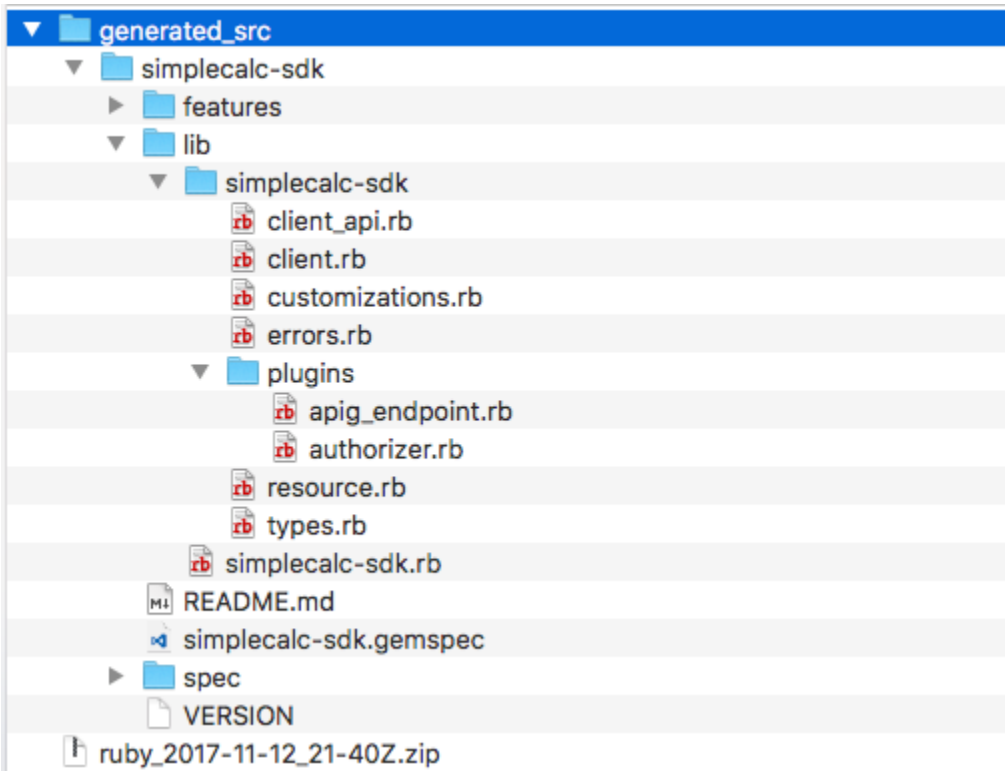
```
1. rootGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the query parameters
2. rootPost(null, body); // where body={"a": ..., "b": ..., "op": ...}
3. aBOpGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the path parameters
```

Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden

> **Note**
>
> Für diese Anleitungen müssen Sie die Anleitungen unter Generieren eines SDK für eine REST-API in API Gateway ausgeführt haben.

So installieren und instanziieren Sie ein von API Gateway generiertes Ruby-SDK für eine REST-API und rufen es auf:

1. Entpacken Sie die heruntergeladene Ruby-SDK-Datei. Die generierte SDK-Quelle wird wie folgt angezeigt.



2. Erstellen Sie mit den folgenden Shell-Befehlen einen Ruby-Gem aus der generierten SDK-Quelle in einem Terminalfenster:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

   Anschließend ist simplecalc-sdk-1.0.0.gem verfügbar.

3. Installieren Sie das Gem:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Erstellen Sie eine Client-Anwendung. Instanziiert und initialisiert den Ruby-SDK-Client in der App:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
```

```
    http_read_timeout: 50
)
```

Wenn die API über eine Autorisierung des konfigurierten `AWS_IAM` Typs verfügt, können Sie die AWS Anmeldeinformationen des Aufrufers angeben, indem Sie `accessKey` und `secretKey` während der Initialisierung angeben:

```
require 'pet-sdk'
client = Pet::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50,
    access_key_id: 'ACCESS_KEY',
    secret_access_key: 'SECRET_KEY'
)
```

5. Führen Sie API-Aufrufe über das SDK in der App durch.

> **Tip**
>
> Wenn Sie nicht mit den Konventionen des SDK-Methodenaufrufs vertraut sind, können Sie die `client.rb`-Datei im generierten `lib`-SDK-Ordner überprüfen. Der Ordner enthält die Dokumentation der einzelnen unterstützten API-Methodenaufrufe.

So erkennen Sie unterstützte Operationen:

```
# to show supported operations:
puts client.operation_names
```

Dies führt entsprechend der API-Methoden `GET /?a={.}&b={.}&op={.}`, `GET /{a}/{b}/{op}`und `POST /` jeweils folgender Anzeige sowie zu einer Nutzlast vom `{a:"…", b:"…", op:"…"}`-Format:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Für den Aufruf der `GET /?a=1&b=2&op=+`-API-Methode rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Für den Aufruf der `POST /`-API-Methode mit einer Nutzlast von `{a: "1", b: "2", "op": "+"}` rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Für den Aufruf der `GET /1/2/+`-API-Methode rufen Sie die folgende Ruby-SDK-Methode auf:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Die erfolgreichen SDK-Methodenaufrufe geben die folgende Antwort zurück:

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

Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden

In diesem Tutorial erfahren Sie, wie Sie ein iOS SDK, das von API Gateway für eine REST-API in einer Objective-C- oder Swift-App generiert wurde, zum Aufrufen der zugrunde liegenden API verwenden. Wir werden die SimpleCalc API als Beispiel verwenden, um die folgenden Themen zu veranschaulichen:

- So installieren Sie die erforderlichen AWS Mobile SDK-Komponenten in Ihrem Xcode-Projekt
- So erstellen Sie das API-Clientobjekt vor dem Aufruf der API-Methoden

- So rufen Sie die API-Methoden mit den entsprechenden SDK-Methoden auf dem API-Clientobjekt auf
- So generieren Sie die Methodeneingabe und analysieren das Ergebnis mit den entsprechenden SDK-Modellklassen

Themen

Verwenden des generierten iOS-SDK (Objective-C) zum Aufrufen der API

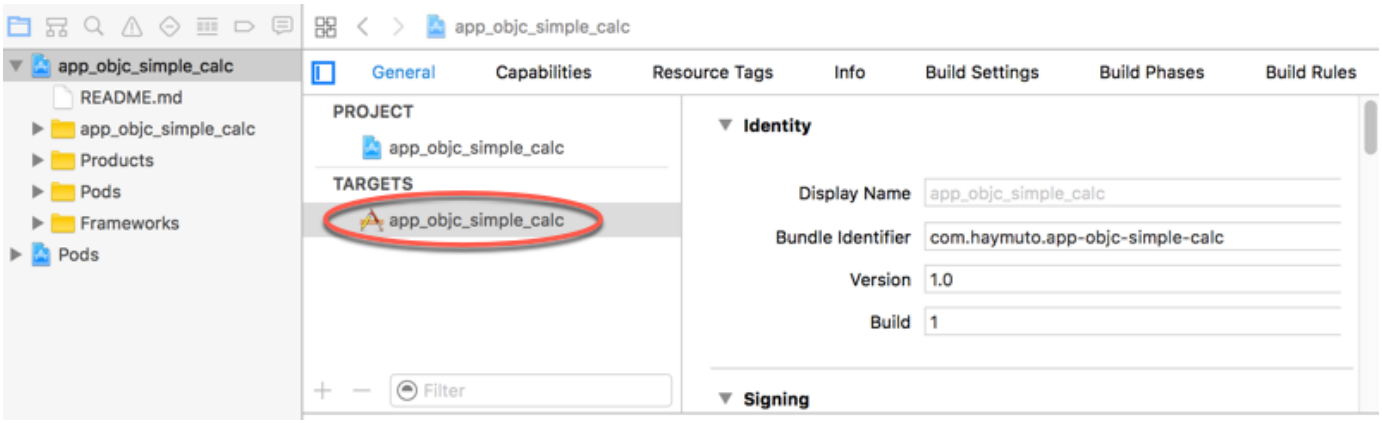
Bevor Sie mit folgendem Verfahren beginnen, müssen Sie die Schritte in Generieren eines SDK für eine REST-API in API Gateway für iOS in Objective-C durchführen und die ZIP-Datei des generierten SDK herunterladen.

Installieren Sie das AWS mobile SDK und ein von API Gateway generiertes iOS-SDK in einem Objective-C-Projekt

Im folgenden Verfahren wird beschrieben, wie Sie das SDK installieren.

So installieren und verwenden Sie ein iOS-SDK, das von API Gateway generiert wurde, in Objective-C:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben. Wenn Sie die SimpleCalc API verwenden, möchten Sie den entpackten SDK-Ordner möglicherweise in etwas wie umbenennen. **`sdk_objc_simple_calc`** In diesem SDK-Ordner sind eine `README.md`-Datei und eine `Podfile`-Datei enthalten. Die `README.md`-Datei enthält die Anweisungen für die Installation und Verwendung des SDK. Dieses Tutorial bietet Details zu diesen Anweisungen. Die Installation nutzt CocoaPodsden Import der erforderlichen API Gateway Gateway-Bibliotheken und anderer abhängiger AWS Mobile SDK-Komponenten. Aktualisieren Sie die `Podfile`, um die SDKs in das Xcode-Projekt Ihrer App zu importieren. Der nicht archivierte SDK-Ordner enthält auch einen `generated-src`-Ordner mit dem Quellcode des generierten SDK Ihrer API.
2. Starten Sie Xcode und erstellen Sie ein neues iOS Objective-C-Projekt. Notieren Sie das Projektziel. Sie müssen die entsprechende Einstellung in der `Podfile` vornehmen.



3. Gehen Sie wie folgt vor, um das mithilfe CocoaPods von AWS Mobile SDK for iOS in das Xcode-Projekt zu importieren:

   a. Führen Sie zur Installation CocoaPods den folgenden Befehl in einem Terminalfenster aus:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Kopieren Sie die `Podfile`-Datei vom extrahierten SDK-Ordner in dasselbe Verzeichnis, das Ihre Xcode Projektdatei enthält. Ersetzen Sie den folgenden Block:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   mit dem Zielnamen Ihres Projekts:

   ```
   target 'app_objc_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Wenn Ihr Xcode-Projekt bereits eine Datei mit dem Namen `Podfile` enthält, fügen Sie die folgende Codezeile hinzu:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

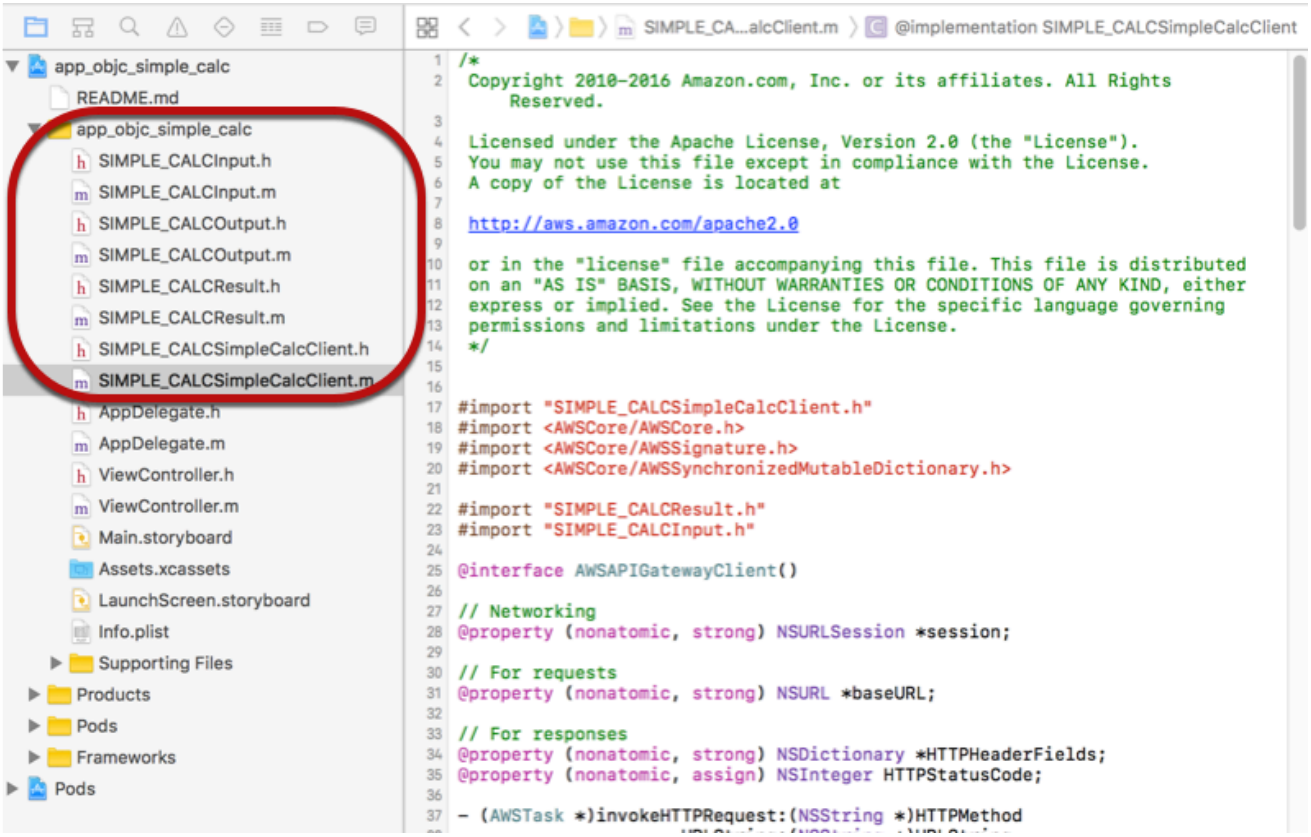
   c. Öffnen Sie ein Terminalfenster und führen Sie den folgenden Befehl aus:

   ```
   pod install
   ```

Dadurch werden die API Gateway Gateway-Komponente und andere abhängige AWS Mobile SDK-Komponenten installiert.

d. Schließen Sie das Xcode-Projekt und öffnen Sie anschließend die `.xcworkspace`-Datei, um Xcode neu zu starten.

e. Fügen Sie alle `.h`- und `.m`-Dateien aus dem extrahierten SDK-Verzeichnis `generated-src` Ihrem Xcode-Projekt hinzu.



Um AWS Mobile SDK for iOS Objective-C in Ihr Projekt zu importieren, indem Sie das AWS Mobile SDK explizit herunterladen oder Carthage verwenden, folgen Sie den Anweisungen in der Datei README.md. Achten Sie darauf, nur eine dieser Optionen zu verwenden, um das Mobile SDK zu importieren. AWS

In einem Objective-C-Projekt API-Methoden unter Verwendung des von API Gateway generierten iOS-SDKs aufrufen

Wenn Sie das SDK mit dem Präfix `SIMPLE_CALC` für diese SimpleCalc API mit zwei Modellen für Eingabe (`Input`) und Ausgabe (`Result`) der Methoden generiert haben, wird im SDK die

resultierende API-Clientklasse `SIMPLE_CALCSimpleCalcClient` und die entsprechenden Datenklassen sind jeweils `SIMPLE_CALCInput` und`SIMPLE_CALCResult`. Die API-Anfragen und-Antworten sind den SDK-Methoden wie folgt zugeordnet:

- Die API-Anfrage

  ```
  GET /?a=...&b=...&op=...
  ```

  wird die SDK-Methode

  ```
  (AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
  ```

  Die Eigenschaft `AWSTask.result` ist vom Typ `SIMPLE_CALCResult`, wenn das Modell `Result` zur Methodenantwort hinzugefügt wurde. Andernfalls ist die Eigenschaft vom Typ `NSDictionary`.
- Diese API-Anfrage

  ```
  POST /

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

  wird die SDK-Methode

  ```
  (AWSTask *)rootPost:(SIMPLE_CALCInput *)body
  ```
- Die API-Anfrage

  ```
  GET /{a}/{b}/{op}
  ```

  wird die SDK-Methode

  ```
  (AWSTask *)aBOpGet:(NSString *)a b:(NSString *)b op:(NSString *)op
  ```

Im folgenden Verfahren wird beschrieben, wie die API-Methoden im Objective-C-App-Quellcode aufgerufen werden, z. B. als Teil des `viewDidLoad`-Delegaten in einer `ViewController.m`-Datei.

So rufen Sie die API über das von API Gateway generierte iOS-SDK auf:

1. Importieren Sie die Header-Datei der API-Client-Klasse, um die API-Client-Klasse in der App aufrufbar zu machen:

   ```
   #import "SIMPLE_CALCSimpleCalc.h"
   ```

   Durch die Anweisung `#import` werden auch `SIMPLE_CALCInput.h` und `SIMPLE_CALCResult.h` für die beiden Modellklassen importiert.

2. Instanziieren Sie die API-Client-Klasse:

   ```
   SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
    defaultClient];
   ```

   Um Amazon Cognito mit der API zu verwenden, legen Sie, wie im Folgenden gezeigt, Sie die `defaultServiceConfiguration`-Eigenschaft auf das Standard-`AWSServiceManager`-Objekt fest, bevor Sie die `defaultClient`-Methode aufrufen, um das API-Client-Objekt zu erstellen (im vorhergehenden Beispiel gezeigt):

   ```
   AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
    initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
   AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
   AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
    configuration;
   ```

3. Rufen Sie die `GET /?a=1&b=2&op=+`-Methode auf, um 1+2 durchzuführen:

   ```
   [[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
    * _Nonnull task) {
       _textField1.text = [self handleApiResponse:task];
       return nil;
   }];
   ```

   in der die Hilfsfunktion `handleApiResponse:task` das Ergebnis als eine Zeichenfolge formatiert, die in einem Textfeld (`_textField1`) angezeigt werden soll.

   ```
   - (NSString *)handleApiResponse:(AWSTask *)task {
       if (task.error != nil) {
   ```

```
        return [NSString stringWithFormat: @"Error: %@", task.error.description];
    } else if (task.result != nil && [task.result isKindOfClass:[SIMPLE_CALCResult
 class]]) {
        return [NSString stringWithFormat:@"%@ %@ %@ = %@\n",task.result.input.a,
 task.result.input.op, task.result.input.b, task.result.output.c];
    }
    return nil;
}
```

Die sich ergebende Anzeige ist `1 + 2 = 3`.

4. Rufen Sie den `POST /` mit einer Nutzlast aus, um `1-2` durchzuführen:

```
SIMPLE_CALCInput *input = [[SIMPLE_CALCInput alloc] init];
    input.a = [NSNumber numberWithInt:1];
    input.b = [NSNumber numberWithInt:2];
    input.op = @"-";
    [[apiInstance rootPost:input] continueWithBlock:^id _Nullable(AWSTask *
 _Nonnull task) {
        _textField2.text = [self handleApiResponse:task];
        return nil;
    }];
```

Die sich ergebende Anzeige ist `1 - 2 = -1`.

5. Rufen Sie `GET /{a}/{b}/{op}` auf, um `1/2` durchzuführen:

```
[[apiInstance aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
 _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

Die sich ergebende Anzeige ist `1 div 2 = 0.5`. Hier wird `div` anstelle von `/` verwendet, da die einfache Lambda-Funktion im Backend keine URL-kodierten Pfadvariablen verarbeitet.

Verwenden eines generierten iOS-SDK (Swift) zum Aufrufen der API

Bevor Sie mit folgendem Verfahren beginnen, müssen Sie die Schritte in Generieren eines SDK für eine REST-API in API Gateway für iOS in Swift durchführen und die ZIP-Datei des generierten SDK herunterladen.

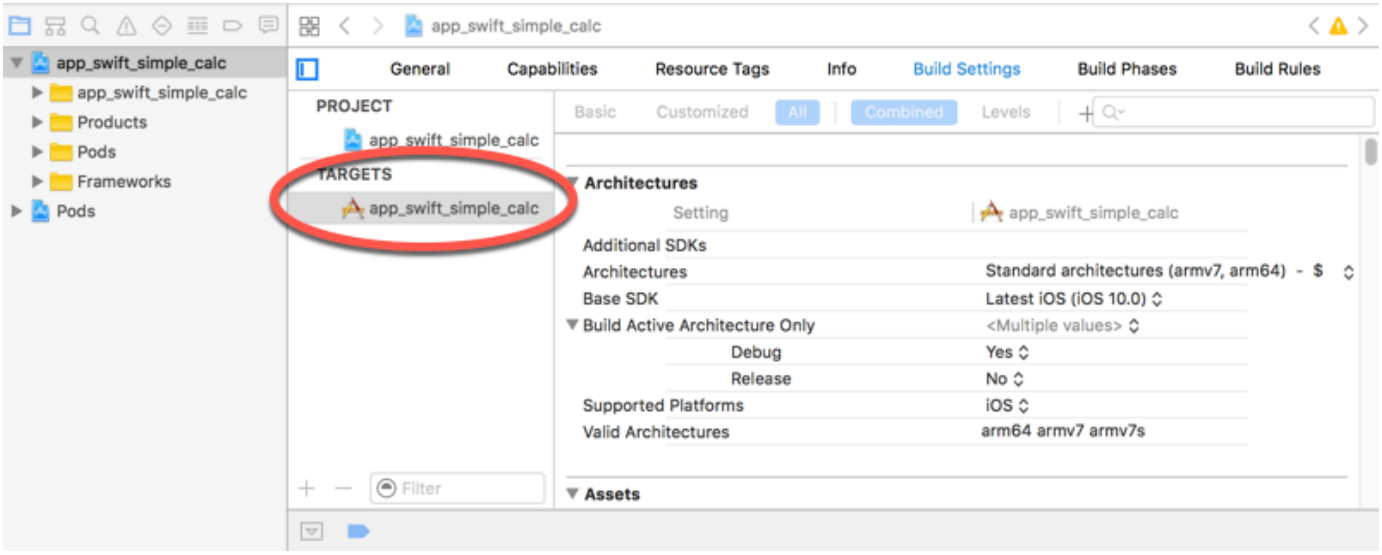
Themen

Installieren Sie das AWS mobile SDK und das vom API Gateway generierte SDK in einem Swift-Projekt

Im folgenden Verfahren wird beschrieben, wie Sie das SDK installieren.

So installieren und verwenden Sie von API Gateway generiertes iOS-SDK in Swift:

1. Extrahieren Sie den Inhalt der von API Gateway generierten ZIP-Datei, die Sie zuvor heruntergeladen haben. Mithilfe der SimpleCalc API möchten Sie den entpackten SDK-Ordner möglicherweise in etwas Ähnliches umbenennen. **`sdk_swift_simple_calc`** In diesem SDK-Ordner sind eine `README.md`-Datei und eine `Podfile`-Datei enthalten. Die `README.md`-Datei enthält die Anweisungen für die Installation und Verwendung des SDK. Dieses Tutorial bietet Details zu diesen Anweisungen. Die Installation nutzt CocoaPodsden Import der erforderlichen AWS Mobile SDK-Komponenten. Aktualisieren Sie die `Podfile`, um die SDKs in das Xcode-Projekt Ihrer Swift-App zu importieren. Der nicht archivierte SDK-Ordner enthält auch einen `generated-src`-Ordner mit dem Quellcode des generierten SDK Ihrer API.
2. Starten Sie Xcode und erstellen Sie ein neues iOS Swift-Projekt. Notieren Sie das Projektziel. Sie müssen die entsprechende Einstellung in der `Podfile` vornehmen.



3. Gehen Sie wie folgt vor, um die erforderlichen AWS Mobile SDK-Komponenten mithilfe CocoaPods von in das Xcode-Projekt zu importieren:

   a. Wenn es nicht installiert ist, installieren Sie es, CocoaPods indem Sie den folgenden Befehl in einem Terminalfenster ausführen:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Kopieren Sie die `Podfile`-Datei vom extrahierten SDK-Ordner in dasselbe Verzeichnis, das Ihre Xcode Projektdatei enthält. Ersetzen Sie den folgenden Block:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   mit dem Zielnamen Ihres Projekts wie dargestellt:

   ```
   target 'app_swift_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Wenn Ihr Xcode-Projekt bereits eine `Podfile` mit dem korrekten Ziel enthält, können Sie einfach die folgende Codezeile zur `do ... end`-Schleife hinzufügen:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

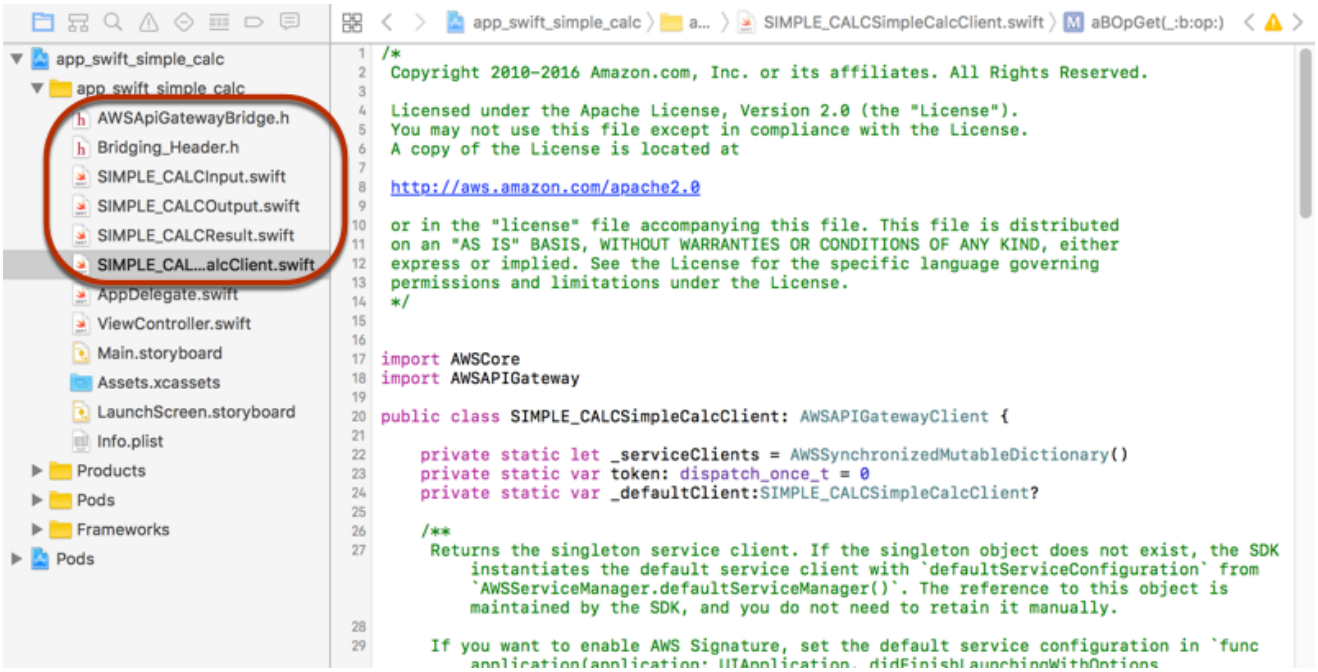
   c. Öffnen Sie ein Terminalfenster und führen Sie den folgenden Befehl im App-Verzeichnis aus:

   ```
   pod install
   ```

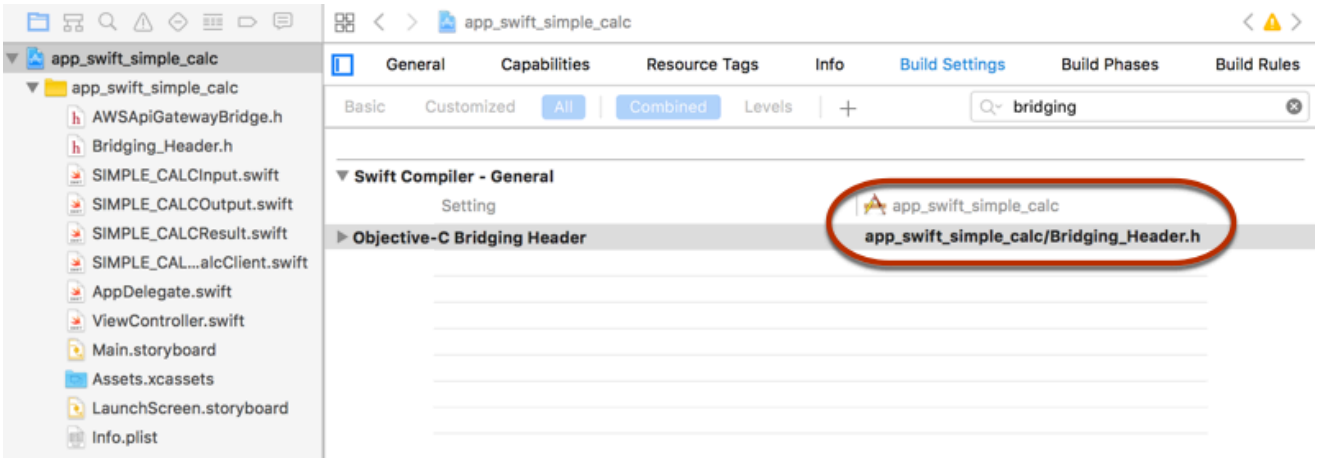
   Dadurch werden die API Gateway Gateway-Komponente und alle abhängigen AWS Mobile SDK-Komponenten im Projekt der App installiert.

   d. Schließen Sie das Xcode-Projekt und öffnen Sie anschließend die `*.xcworkspace`-Datei, um Xcode neu zu starten.

   e. Fügen Sie alle SDK-Header-Dateien (`.h`) und Swift-Quellcodedateien (`.swift`) aus dem extrahierten `generated-src`-Verzeichnis zu Ihrem Xcode-Projekt hinzu.



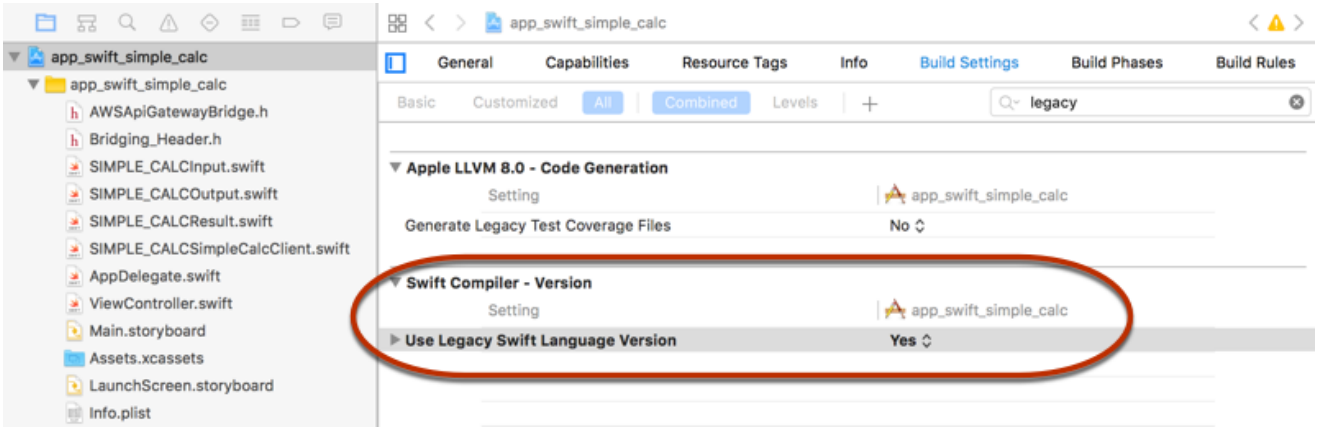
f. Um das Aufrufen der Objective-C-Bibliotheken des AWS Mobile SDK von Ihrem Swift-Codeprojekt aus zu ermöglichen, legen Sie den **`Bridging_Header.h`** Dateipfad in der Eigenschaft Objective-C Bridging Header unter der Einstellung Swift Compiler — General Ihrer Xcode-Projektkonfiguration fest:



> **Tip**
>
> Sie können im Suchfeld von Xcode **`bridging`** eingeben, um die Eigenschaft Objective-C Bridging Header zu suchen.

g. Erstellen Sie das Xcode-Projekt, um zu überprüfen, dass es ordnungsgemäß konfiguriert wurde, bevor Sie fortfahren. Wenn Ihr Xcode eine neuere Version von Swift verwendet als

die, die für das AWS Mobile SDK unterstützt wird, werden Swift-Compilerfehler angezeigt. Setzen Sie in diesem Fall die Eigenschaft Use Legacy Swift Language Version auf Yes in der Einstellung Swift Compiler- Version:



Um das AWS Mobile SDK for iOS in Swift in Ihr Projekt zu importieren, indem Sie das AWS Mobile SDK explizit herunterladen oder Carthage verwenden, folgen Sie den Anweisungen in der `README.md` Datei, die dem SDK-Paket beiliegt. Achten Sie darauf, nur eine dieser Optionen zu verwenden, um das AWS Mobile SDK zu importieren.

API-Methoden über das von API Gateway generierte iOS-SDK in einem Swift-Projekt aufrufen

Wenn Sie das SDK mit dem Präfix `SIMPLE_CALC` für diese SimpleCalc API mit zwei Modellen zur Beschreibung der Eingabe (`Input`) und Ausgabe (`Result`) der Anfragen und Antworten der API generiert haben, wird im SDK die resultierende API-Clientklasse `SIMPLE_CALCSimpleCalcClient` und die entsprechenden Datenklassen sind jeweils `SIMPLE_CALCInput` und`SIMPLE_CALCResult`. Die API-Anfragen und-Antworten sind den SDK-Methoden wie folgt zugeordnet:

- Die API-Anfrage

```
GET /?a=...&b=...&op=...
```

  wird die SDK-Methode

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

  Die Eigenschaft `AWSTask.result` ist vom Typ `SIMPLE_CALCResult`, wenn das Modell `Result` zur Methodenantwort hinzugefügt wurde. Andernfalls ist sie vom Typ `NSDictionary`.

- Diese API-Anfrage

```
POST /

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

  wird die SDK-Methode

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- Die API-Anfrage

```
GET /{a}/{b}/{op}
```

  wird die SDK-Methode

```
public func aBOpGet(a: String, b: String, op: String) -> AWSTask
```

Im folgenden Verfahren wird beschrieben, wie die API-Methoden im Swift-App-Quellcode aufgerufen werden, z. B. als Teil des `viewDidLoad( )`-Delegaten in einer `ViewController.m`-Datei.

So rufen Sie die API über das von API Gateway generierte iOS-SDK auf:

1. Instanziieren Sie die API-Client-Klasse:

   ```
   let client = SIMPLE_CALCSimpleCalcClient.default()
   ```

   Um Amazon Cognito mit der API zu verwenden, legen Sie eine AWS Standard-Servicekonfiguration fest (siehe unten), bevor Sie die `default` Methode aufrufen (siehe oben):

   ```
   let credentialsProvider =
    AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
    "my_pool_id")
   let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
    credentialsProvider: credentialsProvider)
   ```

```
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
 configuration
```

2. Rufen Sie die `GET /?a=1&b=2&op=+`-Methode auf, um 1+2 durchzuführen:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
 in
    self.showResult(task)
    return nil
}
```

in der die Hilfsfunktion `self.showResult(task)` das Ergebnis oder den Fehler auf die Konsole druckt, z. B.:

```
func showResult(task: AWSTask) {
    if let error = task.error {
        print("Error: \(error)")
    } else if let result = task.result {
        if result is SIMPLE_CALCResult {
            let res = result as! SIMPLE_CALCResult
            print(String(format:"%@ %@ %@ = %@", res.input!.a!, res.input!.op!,
 res.input!.b!, res.output!.c!))
        } else if result is NSDictionary {
            let res = result as! NSDictionary
            print("NSDictionary: \(res)")
        }
    }
}
```

In einer Produktion-App können Sie das Ergebnis oder den Fehler in einem Textfeld anzeigen. Die sich ergebende Anzeige ist `1 + 2 = 3`.

3. Rufen Sie den `POST /` mit einer Nutzlast aus, um `1-2` durchzuführen:

```
let body = SIMPLE_CALCInput()
body.a=1
body.b=2
body.op="-"
client.rootPost(body).continueWithBlock {(task: AWSTask) -> AnyObject? in
    self.showResult(task)
    return nil
}
```

Die resultierende Anzeige ist `1 - 2 = -1`.

4. Rufen Sie `GET /{a}/{b}/{op}` auf, um `1/2` durchzuführen:

```
client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
 AnyObject? in
    self.showResult(task)
    return nil
}
```

Die sich ergebende Anzeige ist `1 div 2 = 0.5`. Hier wird `div` anstelle von `/` verwendet, da die einfache Lambda-Funktion im Backend keine URL-kodierten Pfadvariablen verarbeitet.

Konfigurieren einer REST-API mit OpenAPI

Sie können API Gateway verwenden, um eine REST-API aus einer externen Definitionsdatei in API Gateway zu importieren. Gegenwärtig unterstützt API Gateway die Definitionsdateien OpenAPI v2.0 und OpenAPI v3.0, wobei Ausnahmen in Amazon API Gateway – Wichtige Hinweise für REST-APIs aufgeführt werden. Sie können eine API aktualisieren, indem Sie diese mit einer neuen Definition überschreiben, oder Sie können eine Definition mit einer vorhandenen API zusammenführen. Die Optionen in der Anforderungs-URL geben Sie mithilfe eines `mode`-Abfrageparameters an.

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Themen

- Edge-optimierte API in API Gateway importieren
- Importieren Sie eine regionale API in API Gateway
- Importieren einer OpenAPI-Datei zum Aktualisieren einer vorhandenen API-Definition
- Die basePath-Eigenschaft von OpenAPI einrichten
- AWS Variablen für den OpenAPI-Import
- Fehler und Warnungen beim Import
- REST-API von API Gateway importieren

Edge-optimierte API in API Gateway importieren

Geben Sie neben der OpenAPI-Datei für den Importvorgang den Endpunkttyp `EDGE` als zusätzliche Eingabe an, um eine API-OpenAPI-Datei zum Erstellen einer neuen Edge-optimierten API zu importieren. Sie können dies mit der API Gateway Gateway-Konsole oder einem AWS SDK tun. AWS CLI

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Themen

- Edge-optimierte API über die API Gateway-Konsole importieren
- Importieren Sie eine Edge-optimierte API mit dem AWS CLI

Edge-optimierte API über die API Gateway-Konsole importieren

Gehen Sie wie folgt vor, um eine Edge-optimierte API mit der API-Gateway-Konsole zu importieren:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) aus.
3. Unter REST-API, wählen Sie Import (Importieren) aus.
4. Kopieren Sie die OpenAPI-Definition einer API und fügen Sie sie in den Code-Editor ein, oder klicken Sie auf Choose file (Datei auswählen), um eine OpenAPI-Datei von einem lokalen Laufwerk zu laden.
5. Wählen Sie für API-Endpunkt-Typ die Option Edge-optimiert aus.
6. Wählen Sie Create API (API erstellen) aus, um die OpenAPI-Definitionen zu importieren.

Importieren Sie eine Edge-optimierte API mit dem AWS CLI

Um eine API aus einer OpenAPI-Definitionsdatei zu importieren, um mit dem eine neue Edge-optimierte API zu erstellen AWS CLI, verwenden Sie den `import-rest-api` Befehl wie folgt:

```
aws apigateway import-rest-api \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Oder geben Sie für den Abfragezeichenfolgenparameter `endpointConfigurationTypes` ausdrücklich `EDGE` an:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=EDGE \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importieren Sie eine regionale API in API Gateway

Beim Importieren einer API können Sie die regionale Endpunktkonfiguration für die API auswählen. Sie können die API Gateway Gateway-Konsole AWS CLI, das oder ein AWS SDK verwenden.

Die API-Endpunktkonfiguration ist beim Exportieren einer API nicht in den exportierten API-Definitionen enthalten.

Eine Anleitung zur Verwendung der "Import API (API importieren)"-Funktion über die API Gateway-Konsole finden Sie unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Themen

- Regionale API über die API Gateway-Konsole importieren
- Importieren einer regionalen API über die AWS CLI

Regionale API über die API Gateway-Konsole importieren

Gehen Sie wie folgt vor, um eine API eines regionalen Endpunkts über die API Gateway-Konsole zu importieren:

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) aus.
3. Unter REST-API, wählen Sie Import (Importieren) aus.
4. Kopieren Sie die OpenAPI-Definition einer API und fügen Sie sie in den Code-Editor ein, oder klicken Sie auf Choose file (Datei auswählen), um eine OpenAPI-Datei von einem lokalen Laufwerk zu laden.
5. Wählen Sie für API-Endpunkttyp die Option Regional aus.
6. Wählen Sie Create API (API erstellen) aus, um die OpenAPI-Definitionen zu importieren.

Importieren einer regionalen API über die AWS CLI

Verwenden Sie den `import-rest-api` folgenden Befehl, um eine API aus einer OpenAPI-Definitionsdatei mit dem AWS CLI zu importieren:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=REGIONAL \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importieren einer OpenAPI-Datei zum Aktualisieren einer vorhandenen API-Definition

Sie können nur API-Definitionen importieren, um eine vorhandene API zu aktualisieren, ohne die Endpunkt-Konfiguration sowie Phasen und Phasen-Variablen oder Referenzen zu API-Schlüsseln zu verändern.

Der import-to-update Vorgang kann in zwei Modi ausgeführt werden: Zusammenführen oder Überschreiben.

Wenn eine API (`A`) mit einer anderen (`B`) zusammengeführt wird, behält die resultierende API die Definitionen von `A` und `B` bei, wenn die beiden APIs keine widersprüchlichen Definitionen aufweisen. Wenn Konflikte auftreten, überschreiben die Methodendefinitionen der zusammenführenden API (`A`) die entsprechenden Methodendefinitionen der zusammengeführten API (`B`). Beispiel: `B` hat die folgenden Methoden deklariert, um die Antworten `200` und `206` zurückzugeben:

```
GET /a
POST /a
```

und `A` deklariert die folgende Methode, um die Antworten `200` und `400` zurückzugeben:

```
GET /a
```

Wenn `A` mit `B` zusammengeführt wird, ergibt die resultierende API die folgenden Methoden:

```
GET /a
```

die `200`- und `400`-Antworten ausgibt, und

```
POST /a
```

die 200- und 206-Antworten ausgibt.

Die Zusammenführung einer API empfiehlt sich, wenn Sie die externen API-Definitionen in mehrere kleinere Teile zerlegt haben und nur die Änderungen eines dieser Teile anwenden möchten. Dies kann z. B. der Fall sein, wenn für die verschiedenen Teile einer API mehrere Teams zuständig sind, die in unterschiedlichem Tempo Änderungen vornehmen möchten. In diesem Modus bleiben Elemente einer vorhandenen API, die in der importierten Definition nicht speziell definiert wurden, als eigenständige Elemente erhalten.

Wenn eine API (A) eine andere API (B) überschreibt, übernimmt die resultierende API die Definitionen der überschreibenden API (A). Das Überschreiben einer API empfiehlt sich, wenn eine externe API-Definition die vollständige Definition einer API enthält. In diesem Modus werden Elemente einer vorhandenen API, die in der importierten Definition nicht speziell definiert wurden, gelöscht.

Wenn Sie eine API zusammenführen möchten, übermitteln Sie eine PUT-Anforderung an `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. Der Wert `restapi_id` des Parameters "path" gibt die API an, mit der die bereitgestellte API-Definition zusammengeführt wird.

Das folgende Codefragment zeigt ein Beispiel für die PUT-Anfrage zum Zusammenführen einer OpenAPI-Definition in API – als Nutzlast – mit der angegebenen JSON, die sich bereits in API Gateway befindet.

```
PUT /restapis/<restapi_id>?mode=merge
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Bei einer Aktualisierung mittels Zusammenführung werden zwei vollständige API-Definitionen miteinander verbunden. Für geringfügige und inkrementelle Änderungen können Sie die Operation zur Ressourcenaktualisierung verwenden.

Wenn Sie eine API überschreiben möchten, übermitteln Sie eine PUT-Anforderung an `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. Der Wert `restapi_id` des Parameters "path" gibt die API an, die durch die bereitgestellten API-Definitionen überschrieben wird.

Das folgende Codefragment zeigt ein Beispiel für eine Überschreibungsanforderung mit der Nutzlast einer JSON-formatierten OpenAPI-Definition:

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Wenn der Abfrageparameter `mode` fehlt, wird von einer Zusammenführung ausgegangen.

> **Note**
>
> Die PUT-Operationen sind idempotent, aber nicht atomar. Das heißt, wenn während der Verarbeitung ein Systemfehler auftritt, kann die API letztlich einen fehlerhaften Zustand annehmen. Bei erfolgreicher wiederholter Ausführung der Operation nimmt die API jedoch den endgültigen Zustand an, den sie auch bei erfolgreicher Ausführung der ersten Operation erreicht hätte.

Die **`basePath`**-Eigenschaft von OpenAPI einrichten

In OpenAPI 2.0 können Sie mit der `basePath`-Eigenschaft einen oder mehrere Pfadbestandteile bereitstellen, die jedem in den `paths`-Eigenschaften definierten Pfad vorangestellt werden. In API Gateway gibt es mehrere Möglichkeiten, den Pfad einer Ressource auszudrücken daher bietet auch die Funktion zum Importieren von APIs drei Optionen für die Interpretation der `basePath`-Eigenschaft während eines Importvorgangs: Ignorieren, Voranstellen und Splitten.

In OpenAPI 3.0 ist `basePath` keine Top-Level-Eigenschaft mehr. Stattdessen verwendet API Gateway eine Servervariable als Konvention. Die Funktion zum Importieren von APIs bietet die gleichen Optionen für die Interpretation der Basispfades während des Imports. Der Basispfad wird wie folgt identifiziert:

- Wenn die API keine `basePath`-Variablen aufweist, überprüft die Funktion zum Importieren von APIs, ob die `server.url`-Zeichenfolge einen Pfad außer `"/"` enthält. Wenn dies der Fall ist, wird dieser Pfad als Basispfad verwendet.

- Wenn die API nur eine `basePath`-Variable enthält, verwendet die Funktion zum Importieren von APIs diese als Basispfad, auch wenn sie nicht in `server.url` referenziert wird.
- Wenn die API mehrere `basePath`-Variablen enthält, verwendet die Funktion zum Importieren von APIs nur den ersten Eintrag als Basispfad.

Ignore

Wenn in der OpenAPI-Datei der `basePath`-Wert `/a/b/c` lautet und die `paths`-Eigenschaft `/e` und `/f`enthält, führt die folgende `POST`- oder `PUT`-Anforderung:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/e`
- `/f`

Dadurch wird die `basePath`-Eigenschaft so behandelt, als sei sie nicht vorhanden, und alle deklarierten API-Ressourcen werden relativ zum Host bereitgestellt. Dies empfiehlt sich z. B., wenn Sie einen benutzerdefinierten Domänennamen mit einem API-Mapping ohne einen Wert für den Basispfad und mit einem Wert für die Phase (die sich auf Ihre Produktionsphase bezieht) haben.

> **Note**
>
> API Gateway erstellt automatisch eine Root-Ressource für Sie, auch wenn diese nicht explizit in Ihrer Definitionsdatei deklariert ist.

Wenn nicht angegeben, wird `basePath` standardmäßig auf `ignore` gesetzt.

Voranstellen

Wenn in der OpenAPI-Datei der Wert der `basePath`-Eigenschaft lautet und die `/a/b/c`-Eigenschaft `paths`und`/e` enthält, führt die folgende `/f``POST`- oder `POSTPUTPUT`-Anforderung:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

Dadurch wird die Eigenschaft `basePath` so behandelt, als würde sie zusätzliche Ressourcen (ohne Methoden) angeben und diese zum deklarierten Ressourcensatz hinzufügen. Dies empfiehlt sich z. B., wenn verschiedene Teams für die verschiedene Teile einer API zuständig sind und die Eigenschaft `basePath` auf den Pfadpfadspeicherort des API-Teils der einzelnen Teams verweisen kann.

> **Note**
>
> API Gateway erstellt automatisch Zwischenstufen von Ressourcen für Sie, auch wenn Sie diese nicht ausdrücklich in der Definition deklariert haben.

Teilen

Wenn in der OpenAPI-Datei der Wert der `basePath`-Eigenschaft lautet und die `/a/b/c`-Eigenschaft `paths`und`/e` enthält, führt die folgende `/f`POST- oder `POSTPUT`PUT-Anforderung:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

führt dies zu den folgenden Ressourcen in der API:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

Dadurch wird die oberste Ebene des Pfades (`/a`) als Beginn aller Pfade der Ressource behandelt und es werden zusätzliche Ressourcen (keine Methoden) in der API an sich erstellt. Das empfiehlt sich z. B., wenn a ein Phasenname ist, den Sie als Teil Ihrer API bereitstellen möchten.

AWS Variablen für den OpenAPI-Import

Sie können die folgenden AWS Variablen in OpenAPI-Definitionen verwenden. API Gateway löst die Variablen auf, wenn die API importiert wird. Um eine Variable anzugeben, verwenden Sie `${`*`variable-name`*`}`.

AWS Variablen

Variablenname	Beschreibung
`AWS::AccountId`	Die AWS Konto-ID, die die API importiert, z. B. 123456789012.
`AWS::Partition`	Die AWS Partition, in die die API importiert wird. Für AWS Standardregionen ist die Partition `aws`.
`AWS::Region`	Die AWS Region, in die die API importiert wird — zum Beispiel. `us-east-2`

AWS Beispiel für Variablen

Im folgenden Beispiel AWS werden Variablen verwendet, um eine AWS Lambda Funktion für eine Integration anzugeben.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "tasks-api"
  version: "v1.0"
paths:
  /:
    get:
      summary: List tasks
      description: Returns a list of tasks
      responses:
        200:
          description: "OK"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Task"
        500:
          description: "Internal Server Error"
          content: {}
      x-amazon-apigateway-integration:
        uri:
          arn:${AWS::Partition}:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/arn:${AWS::Partition}:lambda:${AWS::Region}:
${AWS::AccountId}:function:LambdaFunctionName/invocations
        responses:
          default:
            statusCode: "200"
        passthroughBehavior: "when_no_match"
        httpMethod: "POST"
        contentHandling: "CONVERT_TO_TEXT"
        type: "aws_proxy"
components:
  schemas:
    Task:
      type: object
      properties:
        id:
          type: integer
        name:
          type: string
```

```
        description:
          type: string
```

Fehler und Warnungen beim Import

Fehler beim Import

Während des Imports werden für schwerwiegende Probleme wie ungültige OpenAPI-Dokumente Fehler generiert. Fehler werden in Form von Ausnahmen als Antwort auf eine erfolglose Aktion zurückgegeben (z. B. `BadRequestException`). Wenn ein Fehler auftritt, wird die neue API-Definition verworfen und die vorhandene API wird unverändert beibehalten.

Warnungen beim Import

Während des Imports werden für kleinere Probleme wie fehlende Modellreferenzen Warnungen generiert. Falls eine Warnung auftritt, wird der Vorgang fortgesetzt, wenn an die Anforderungs-URL der Abfrageausdruck `failonwarnings=false` angehängt wird. Andernfalls werden die Aktualisierungen rückgängig gemacht. `failonwarnings` ist standardmäßig auf `false` festgelegt. In solchen Fällen werden Warnungen als Feld in der resultierenden RestApiRessource zurückgegeben. Andernfalls werden Warnungen als Nachricht in der Ausnahme ausgegeben.

REST-API von API Gateway importieren

Sobald Sie eine REST-API in API Gateway erstellt und konfiguriert haben, können Sie sie mit der API Gateway-Konsole oder anderweitig in eine OpenAPI-Datei exportieren, und zwar mit der API Gateway-Export-API, die Teil des Amazon API Gateway-Control-Service ist. Um die API-Gateway-Export-API verwenden zu können, müssen Sie Ihre API-Anforderungen signieren. Weitere Informationen zum Signieren von Anfragen finden Sie unter Signieren von AWS API-Anfragen im IAM-Benutzerhandbuch. Ihnen stehen Optionen zur Verfügung, die API Gateway-Integrationserweiterungen sowie die Postman-Erweiterungen in die exportierte OpenAPI-Definitionsdatei aufzunehmen.

> **Note**
>
> Achten Sie beim Exportieren der API mithilfe von darauf AWS CLI, den Erweiterungsparameter anzugeben, wie im folgenden Beispiel gezeigt, um sicherzustellen, dass die `x-amazon-apigateway-request-validator` Erweiterung enthalten ist:

```
aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
 abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
```

Sie können eine API nicht exportieren, wenn deren Nutzlast nicht vom Typ `application/json` ist. Wenn Sie dies versuchen, erhalten Sie eine Fehlermeldung, die besagt, dass JSON-Textmodelle nicht gefunden wurden.

Anfordern zum Exportieren einer REST-API

Mit der Export-API exportieren Sie eine vorhandene REST-API, indem Sie eine GET-Anfrage einreichen und die to-be-exported API als Teil der URL-Pfade angeben. Der Anforderungs-URL hat das folgende Format:

OpenAPI 3.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/oas30
```

OpenAPI 2.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/swagger
```

Sie können die Abfragezeichenfolge `extensions` anhängen, um anzugeben, ob API Gateway-Erweiterungen (mit dem Wert `integration`) oder Postman-Erweiterungen (mit dem Wert `postman`) enthalten sein sollen.

Darüber hinaus können Sie den Header `Accept` auf `application/json` oder `application/yaml` festlegen, um die API-Definitionsausgabe im JSON- bzw. YAML-Format zu erhalten.

Weitere Informationen zum Einreichen von GET-Anfragen mit der API Gateway Export API finden Sie unter GetExport.

> **Note**
>
> Wenn Sie Modelle in Ihrer API definieren, müssen diese für den Inhaltstyp "application/json" sein, damit API Gateway das Modell exportieren kann. Andernfalls löst API Gateway eine Ausnahme mit der Fehlermeldung "Only found non-JSON body models for ..." aus. Modelle müssen Eigenschaften enthalten oder als ein bestimmter JSONSchema-Typ definiert werden.

Herunterladen der REST-API-OpenAPI-Definition als JSON

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen im JSON-Format herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Hier kann *<region>* beispielsweise `us-east-1` sein. Informationen zu allen Regionen, in denen API Gateway verfügbar ist, finden Sie unter Regionen und Endpunkte.

Herunterladen der REST-API-OpenAPI-Definition als YAML

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen im YAML-Format herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Herunterladen der REST-API-OpenAPI-Definition mit Postman-Erweiterungen als JSON

So exportieren und laden Sie eine REST-API in OpenAPI-Definitionen mit Postman im JSON-Format herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

REST-API OpenAPI-Definition mit API Gateway-Integration in YAML herunterladen

So exportieren Sie eine REST-API in OpenAPI-Definitionen mit API Gateway-Integration im YAML-Format und laden Sie herunter:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

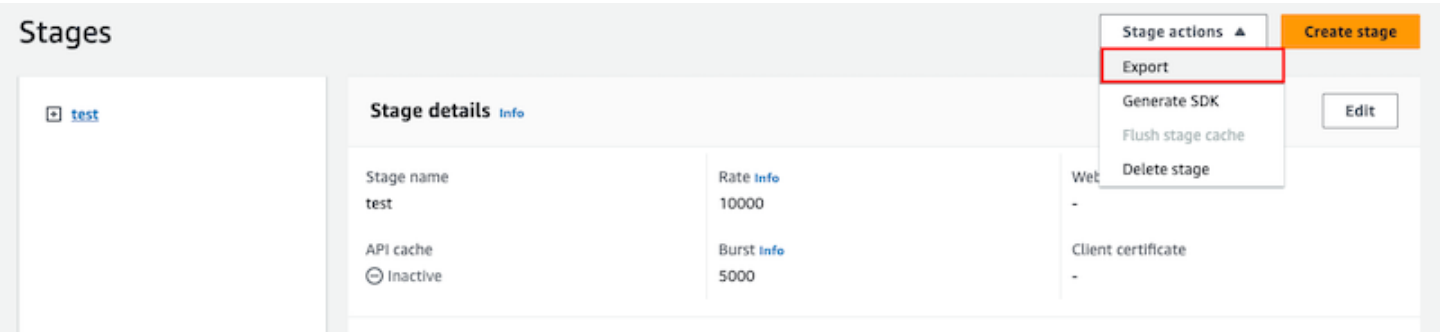
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

REST-API über die API-Gateway-Konsole exportieren

Nach der Bereitstellung Ihrer REST-API für eine Stufe können Sie mit der API Gateway-Konsole die API in der Stufe in eine OpenAPI-Datei exportieren.

Wählen Sie in der API-Gateway-Konsole im Bereich Stufen die Optionen Stufenaktionen und Export aus.



Geben Sie einen API-Spezifikationstyp, ein Format und Erweiterungen an, um die OpenAPI-Definition Ihrer API herunterzuladen.

Veröffentlichen von REST-APIs, mit denen Kunden aufgerufen werden können

Das einfache Erstellen und Entwickeln einer API Gateway-API macht sie nicht automatisch für Ihre Benutzer aufrufbar. Um sie aufrufbar zu machen, müssen Sie Ihre API in einer Stufe bereitstellen. Darüber hinaus können Sie die URL anpassen, die Ihre Benutzer für den Zugriff auf Ihre API verwenden. Sie können ihr eine Domäne zuweisen, die mit Ihrer Marke übereinstimmt oder einprägsamer ist als die Standard-URL für Ihre API.

In diesem Abschnitt erfahren Sie, wie Sie Ihre API bereitstellen und die URL anpassen, die Sie Benutzern für den Zugriff zur Verfügung stellen.

> **Note**
>
> Um die Sicherheit Ihrer API-Gateway-APIs zu erhöhen, ist die `execute-api.{region}.amazonaws.com`-Domain in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-`-Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domain-Namen für Ihre API-Gateway-APIs einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Themen

- Bereitstellen einer REST-API in Amazon API Gateway
- Einrichten von benutzerdefinierten Domänennamen für REST-APIs

Bereitstellen einer REST-API in Amazon API Gateway

Nach der Erstellung der API müssen Sie diese bereitstellen, damit sie von den Benutzern aufgerufen werden kann.

Zum Bereitstellen einer API erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Stufe. Eine Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B, `dev`, `prod`, `beta`, `v2`). API-Stufen werden durch die API-ID und den Stufennamen identifiziert. Sie sind in der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

> **Important**
>
> Jedes Mal, wenn Sie eine API aktualisieren, müssen Sie die API in einer vorhandenen Stufe oder in einer neuen Stufe erneut bereitstellen. Das Aktualisieren einer API umfasst das Ändern von Routen, Methoden, Integrationen, Autorisierern, Ressourcenrichtlinien und allem anderen als den Stufeneinstellungen.

Wenn sich Ihre API erweitert, können Sie sie als unterschiedliche Versionen für verschiedene Stufen bereitstellen. Sie können Ihre API-Updates auch als Canary-Release-Bereitstellung bereitstellen. Auf diese Weise können Ihre API-Clients auf derselben Stufe über die Produktionsveröffentlichung auf die Produktionsversion und über die Canary-Veröffentlichung auf die aktualisierte Version zugreifen.

Zum Aufrufen einer bereitgestellten API sendet der Client eine Anforderung mit einer API-URL. Die URL wird anhand eines API-Protokolls (HTTP (S) oder (WSS)), Hostnamens, Stufennamens und (für REST-APIs) Ressourcen-Pfads bestimmt. Der Hostname und der Stufenname bestimmen die Basis-URL der API.

Unter Verwendung des Standarddomänennamens der API weist die Basis-URL einer REST-API (beispielsweise) in einer bestimmten Stufe (*`{stageName}`*) das folgende Format auf:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Sie können einen benutzerdefinierten Domänennamen (z. B. `api.example.com`) erstellen und damit den standardmäßigen Hostnamen der API ersetzen, um die Basis-URL der API benutzerfreundlicher zu gestalten. Um mehrere APIs unter dem benutzerdefinierten Domänennamen zu unterstützen, müssen Sie einem Basispfad eine API-Stufe zuordnen.

Mit dem benutzerdefinierten Domänennamen *`{api.example.com}`* und der einem Basispfad (*`{basePath}`*) unter dem benutzerdefinierten Domänennamen zugeordneten API-Stufe, ändert sich die Basis-URL einer REST-API wie folgt:

```
https://{api.example.com}/{basePath}
```

Sie können für jede Stufe die API-Leistung optimieren, indem Sie die standardmäßigen Drosselungslimits für Anforderungen auf Kontoebene anpassen und das API-Caching aktivieren. Sie können auch die Protokollierung für API-Aufrufe an CloudTrail oder aktivieren und ein Client-Zertifikat für das Backend auswählen CloudWatch, um die API-Anfragen zu authentifizieren. Außerdem können Sie die Stufeneinstellungen für einzelne Methoden überschreiben und Stufenvariablen definieren, um stufenbezogenen Umgebungskontext zur Laufzeit an die API-Integration zu übergeben.

Stufen ermöglichen eine robuste Versionskontrolle Ihrer API. Sie können beispielsweise eine API für eine `test`- und eine `prod`-Stufe bereitstellen und die `test`-Stufe als Test-Build und die `prod`-Stufe als stabilen Build verwenden. Nachdem die Aktualisierungen den Test bestanden haben, können Sie die `test`-Stufe auf `prod` hochstufen. Die Hochstufung kann durch die erneute Bereitstellung der API auf der `prod`-Stufe oder durch Aktualisieren eines Stufenvariablenwerts vom Stufennamen `test` auf `prod` erfolgen.

In diesem Abschnitt wird die Bereitstellung einer API mithilfe der API Gateway-Konsole oder durch Aufrufen der API Gateway-REST-API beschrieben. Informationen zur Verwendung anderer Tools finden Sie in der Dokumentation der AWS CLI oder eines AWS SDK.

Themen

- Bereitstellen einer REST-API in API Gateway
- Einrichten einer Stufe für eine REST-API
- Einrichten einer API Gateway-Canary-Release-Bereitstellung
- Aktualisierungen für eine REST-API, die eine erneute Bereitstellung erfordern

Bereitstellen einer REST-API in API Gateway

In API Gateway wird eine REST-API-Bereitstellung durch eine Deployment-Ressource dargestellt. Sie ähnelt einer ausführbaren Datei einer API, die durch eine RestApi-Ressource dargestellt wird.

Damit der Client Ihre API aufrufen kann, müssen Sie eine Bereitstellung erstellten und ihr eine Stufe zuordnen. Eine Stufe wird durch eine Stage-Ressource dargestellt. Sie stellt einen Snapshot der API dar, einschließlich Methoden, Integrationen, Modellen, Mapping-Vorlagen und Lambda-Genehmigern (ehemals als benutzerdefinierte Genehmiger bezeichnet). Wenn Sie die API aktualisieren, können Sie sie erneut bereitstellen, indem Sie der vorhandenen Bereitstellung eine neue Stufe zuordnen. Das Erstellen einer Stufe wird in beschriebe the section called “Einrichten einer Stufe”.

Themen

Erstellen einer Bereitstellung mit der AWS CLI

Wenn Sie eine Bereitstellung erstellen, instanziieren Sie die Deployment-Ressource. Sie können die API-Gateway-Konsole, die AWS CLI, ein AWS SDK oder die API-Gateway-REST-API verwenden, um eine Bereitstellung zu erstellen.

Zum Erstellen einer Bereitstellung mit der CLI verwenden Sie den Befehl create-deployment:

```
aws apigateway create-deployment --rest-api-id <rest-api-id> --region <region>
```

Die API kann erst aufgerufen werden, wenn Sie dieser Bereitstellung eine Stufe zugeordnet haben. Diesen Vorgang können Sie bei einer bereits vorhandenen Stufe durch Aktualisieren der Eigenschaft `deploymentId` der Stufe mit der neu erstellten Bereitstellungs-ID (`<deployment-id>`) ausführen.

```
aws apigateway update-stage --region <region> \
   --rest-api-id <rest-api-id> \
   --stage-name <stage-name> \
   --patch-operations op='replace',path='/deploymentId',value='<deployment-id>'
```

Bei der ersten Bereitstellung einer API können Sie die Stufe und die Bereitstellung gleichzeitig erstellen:

```
aws apigateway create-deployment --region <region> \
   --rest-api-id <rest-api-id> \
   --stage-name <stage-name>
```

Dieser Vorgang wird im Hintergrund in der API Gateway-Konsole ausgeführt, wenn Sie eine API zum ersten Mal oder erneut für eine neue Stufe bereitstellen.

Bereitstellen einer REST-API über die API Gateway-Konsole

Sie müssen eine REST API erstellt haben, bevor Sie sie erstmals bereitstellen können. Weitere Informationen finden Sie unter REST-API in API Gateway entwickeln.

Themen

Bereitstellen einer REST-API für eine Stufe

Mit der API Gateway-Konsole können Sie eine API bereitstellen, indem Sie eine Bereitstellung erstellen und sie mit einer neuen oder bestehenden Stufe verknüpfen.

> **Note**
>
> Informationen zum Verknüpfen einer Stufe in API Gateway mit einer anderen Bereitstellung finden Sie stattdessen unter Zuweisen einer Stufe mit einer anderen REST-API-Bereitstellung.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie im APIs-Navigationsbereich die API, die Sie bereitstellen möchten.
3. Wählen Sie im Bereich Ressourcen die Option API bereitstellen aus.
4. Wählen Sie für Stufe eine der folgenden Optionen aus:
   a. Um eine neue Stufe zu erstellen, wählen Sie Neue Stufe aus und geben Sie dann einen Namen in das Feld Stufenname ein. Optional können Sie unter Beschreibung der Bereitstellung eine Beschreibung für die Bereitstellung angeben.
   b. Um eine bestehende Stufe auszuwählen, wählen Sie den Namen der Stufe im Drop-down-Menü aus. Sie sollten unter Beschreibung der Bereitstellung eine Beschreibung der neuen Bereitstellung angeben.
   c. Um eine Bereitstellung zu erstellen, die keiner Stufe zugeordnet ist, wählen Sie Keine Stufe aus. Später können Sie diese Bereitstellung einer Stufe zuordnen.
5. Wählen Sie Bereitstellen.

Erneutes Bereitstellen einer REST-API für eine Stufe

Um eine API erneut bereitzustellen, führen Sie die gleichen Schritte wie in au the section called “Bereitstellen einer REST-API für eine Stufe”. Sie können dieselbe Stufe so oft wie gewünscht verwenden.

Aktualisieren der Stufenkonfiguration einer REST-API-Bereitstellung

Nachdem eine API bereitgestellt wurde, können Sie die Stufeneinstellungen ändern, um den API-Cache, die Protokollierung oder die Anforderungsdrosselung zu aktivieren oder zu deaktivieren. Sie können auch ein Client-Zertifikat wählen, damit das Backend API Gateway authentifizieren und Stufenvariablen festlegen kann, um den Bereitstellungskontext während der Laufzeit an die API-Integration zu übergeben. Weitere Informationen finden Sie unter Aktualisieren von Stufeneinstellungen.

> **Important**
>
> Nachdem Sie die Stufeneinstellungen geändert haben, müssen Sie die API erneut bereitstellen, damit die Änderungen wirksam werden.

> **Note**
>
> Wenn die aktualisierten Einstellungen, wie die Aktivierung der Protokollierung, eine neue IAM-Rolle erfordern, können Sie die erforderliche IAM-Rolle ohne erneute Bereitstellung der API hinzufügen. Allerdings kann es einige Minuten dauern, bis die neue IAM-Rolle wirksam wird. Bevor dies jedoch der Fall ist, werden Spuren Ihrer API-Aufrufe nicht protokolliert, auch wenn Sie die Protokollierung aktiviert haben.

Einrichten von Stufenvariablen für eine REST-API-Bereitstellung

Für eine Bereitstellung können Sie Stufenvariablen so festlegen oder ändern, dass bereitstellungsspezifische Daten während der Laufzeit an die API-Integration weitergeben werden. Sie können dies auf der Registerkarte Stage Variables im Stage Editor tun. Weitere Informationen finden Sie in den Anweisungen in Einrichten von Stufenvariablen für eine REST-API-Bereitstellung.

Zuweisen einer Stufe mit einer anderen REST-API-Bereitstellung

Da eine Bereitstellung einen API-Snapshot darstellt und eine Stufe einen Pfad in einen Snapshot definiert, können Sie verschiedene Bereitstellungsstufen-Kombinationen wählen, um zu steuern, wie Benutzer verschiedene Versionen der API aufrufen. Dies ist beispielsweise nützlich, wenn Sie den API-Status auf eine frühere Bereitstellung zurücksetzen oder eine "private Zweigstelle" der API in eine öffentliche integrieren möchten.

Im folgenden Verfahren wird gezeigt, wie Sie dies mithilfe des Stage Editor (Stufen-Editor) in der API Gateway-Konsole tun. Es wird davon ausgegangen, dass Sie eine API mehr als einmal bereitgestellt haben.

1. Wenn Sie sich noch nicht im Bereich Stufen befinden, wählen Sie im Hauptnavigationsbereich die Option Stufen aus.
2. Wählen Sie die Stufe aus, die Sie aktualisieren möchten.
3. Wählen Sie auf der Registerkarte Bereitstellungsverlauf die Bereitstellung aus, die die Stufe verwenden soll.
4. Wählen Sie Aktive Bereitstellung ändern.
5. Bestätigen Sie, dass Sie die aktive Bereitstellung ändern möchten, und wählen Sie Aktive Bereitstellung ändern im Dialogfeld Bereitstellung aktivieren aus.

Einrichten einer Stufe für eine REST-API

Eine Stufe ist eine benannte Referenz auf eine Bereitstellung, bei der es sich um einen Snapshot der API handelt. Sie verwenden eine Stufe, um eine bestimmte Bereitstellung zu verwalten und zu optimieren. Beispielsweise können Sie Stufeneinstellungen konfigurieren, um Zwischenspeicherung zu aktivieren, Anforderungsdrosselung anzupassen, Protokollierung zu konfigurieren, Stufenvariablen zu definieren oder ein Canary-Release für Tests anzufügen.

Themen

Einrichten einer Stufe mithilfe der API Gateway-Konsole

Themen

Erstellen einer neuen Stufe

Nach der ersten Bereitstellung können Sie weitere Stufen hinzufügen und sie mit vorhandenen Bereitstellungen verknüpfen. Sie können über die API Gateway-Konsole eine neue Stufe erstellen oder während der Bereitstellung einer API eine vorhandene Stufe auswählen. Im Allgemeinen fügen Sie eine neue Stufe zu einer API-Bereitstellung vor der erneuten Bereitstellung der API hinzu. Gehen Sie folgendermaßen vor, um mithilfe der API Gateway-Konsole eine neue Stufe zu erstellen:

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich unter APIs die Option Stages (Stufen).
4. Wählen Sie im Navigationsbereich Stages (Stufen) die Option Create stage (Stufe erstellen).
5. Geben Sie unter Stage name (Stufenname) einen Namen ein (z. B. **`prod`**).

   > **Note**
   >
   > Phasennamen dürfen nur alphanumerische Zeichen sowie Binde- und Unterstriche enthalten. Die maximale Länge beträgt 128 Zeichen.

6. (Optional). Geben Sie unter Description (Beschreibung) eine kurze Beschreibung der Stufe ein.
7. Wählen Sie in der Dropdown-Liste Deployment (Bereitstellung) das Datum und die Uhrzeit der vorhandenen API-Bereitstellung aus, die Sie dieser Stufe zuordnen möchten.
8. Unter Zusätzliche Einstellungen können Sie zusätzliche Einstellungen für Ihre Stufe angeben.
9. Wählen Sie Create stage (Stufe erstellen) aus.

Aktualisieren von Stufeneinstellungen

Nach der erfolgreichen Bereitstellung einer API wird die Stufe mit Standardeinstellungen gefüllt. Sie können die Konsole oder die API Gateway-REST-API verwenden, um die Stufeneinstellungen zu

ändern, einschließlich des API-Cachings und der Protokollierung. Im Folgenden wird gezeigt, wie Sie dies mithilfe des Stufen-Editor der API Gateway-Konsole durchführen.

Aktualisieren der Stufeneinstellungen mithilfe der API Gateway-Konsole

Diese Schritte gehen davon aus, dass Sie die API bereits in einer Stufe bereitgestellt haben.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich unter APIs die Option Stages (Stufen).
4. Wählen Sie im Bereich Stages den Namen der Stufe aus.
5. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
6. (Optional) Bearbeiten Sie die Beschreibung in Stufenbeschreibung.
7. Ändern Sie die folgenden Einstellungen in Zusätzliche Einstellungen:

   Cache-Einstellungen

   Um API-Caching für die Phase zu aktivieren, aktivieren Sie API-Cache bereitstellen. Konfigurieren Sie dann das Standard-Caching auf Methodenebene, die Cachekapazität, die Cache-Daten verschlüsseln, den Cache time-to-live (TTL) sowie alle Anforderungen für die Cache-Invalidierung pro Schlüssel.

   Das Caching ist erst aktiv, wenn Sie das standardmäßige Caching auf Methodenebene oder den Cache auf Methodenebene für eine bestimmte Methode aktivieren.

   Weitere Informationen zu Cache-Einstellungen finden Sie unter Aktivieren von API-Caching für verbesserte Reaktionsfähigkeit.

   > **Note**
   >
   > Wenn Sie API-Caching für eine API-Phase aktivieren, wird Ihr AWS Konto möglicherweise für das API-Caching belastet. Caching kommt nicht für das kostenlose Kontingent in Frage. AWS

Drosselungseinstellungen

Aktivieren Sie die Option Drosselung zum Einrichten der Drosselungs-Targets auf Stufenebene für alle Methoden, die dieser API zugeordnet sind.

Geben Sie für Rate einen Zielsatz ein. Dies ist die Rate in Anfragen pro Sekunde, mit der Token zum Token-Bucket hinzugefügt werden. Diese Rate auf Stufenebene darf nicht größer sein als die Rate auf Kontoebene, die unter API Gateway-Kontingente für die Konfiguration und Ausführung einer REST-API vorgegeben ist.

Geben Sie für Burst eine Ziel-Burst-Rate ein. Die Burst-Rate ist die Kapazität des Token-Buckets. Dies ermöglicht mehr Anfragen für einen bestimmten Zeitraum als der Zielsatz. Diese Burst-Rate auf Stufenebene darf nicht größer sein als die Burst-Rate auf Kontoebene, die unter API Gateway-Kontingente für die Konfiguration und Ausführung einer REST-API angegeben ist.

> **Note**
>
> Drosselungsraten sind keine harten Grenzwerte und werden auf Best-Effort-Basis angewendet. In manchen Fällen können Kunden die von Ihnen festgelegten Ziele überschreiten. Verlassen Sie sich nicht auf die Drosselung, um Kosten zu kontrollieren oder den Zugriff auf eine API zu blockieren. Ziehen Sie es in Betracht, AWS Budgets zu verwenden, um Kosten zu überwachen und AWS WAF, um API-Anfragen zu verwalten.

Firewall- und Zertifikatseinstellungen

Um der Phase eine AWS WAF Web-ACL zuzuordnen, wählen Sie eine Web-ACL aus der Dropdownliste Web-ACL aus. Falls gewünscht, wählen Sie Block API Request if WebACL cannot be evaluated (Fail- Close) (API-Anforderung blockieren, wenn WebACL nicht ausgewertet werden kann (Fehler – Schließen)).

Um ein Client-Zertifikat für Ihre Stufe auszuwählen, wählen Sie ein Zertifikat aus dem Dropdownmenü Client certificate (Client-Zertifikat) aus.

8. Wählen Sie Speichern.

9. Um Amazon CloudWatch Logs für alle Methoden zu aktivieren, die mit dieser Phase dieser API-Gateway-API verknüpft sind, wählen Sie im Abschnitt Logs and tracing die Option Bearbeiten.

> **Note**
>
> Um CloudWatch Logs zu aktivieren, müssen Sie auch den ARN einer IAM-Rolle angeben, die es API Gateway ermöglicht, im Namen Ihres Benutzers Informationen in CloudWatch Logs zu schreiben. Wählen Sie dazu Settings im Hauptnavigationsbereich APIs. Geben Sie dann für die CloudWatch Protokollrolle den ARN einer IAM-Rolle ein. In gängigen Anwendungsszenarien kann die IAM-Rolle beispielsweise die verwaltete Richtlinie von `AmazonAPIGatewayPushToCloudWatchLogs` anfügen, die die folgende Zugriffsrichtlinienanweisung enthält:
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>         {
>             "Effect": "Allow",
>             "Action": [
>                 "logs:CreateLogGroup",
>                 "logs:CreateLogStream",
>                 "logs:DescribeLogGroups",
>                 "logs:DescribeLogStreams",
>                 "logs:PutLogEvents",
>                 "logs:GetLogEvents",
>                 "logs:FilterLogEvents"
>             ],
>             "Resource": "*"
>         }
>     ]
> }
> ```
>
> Die IAM-Rolle muss zudem die folgende Vertrauensstellungsanweisung enthalten:
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Sid": "",
>       "Effect": "Allow",
>       "Principal": {
> ```

```
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

10. Wählen Sie im Dropdownmenü Protokolle eine CloudWatch Protokollierungsebene aus. Die Protokollierungsebenen sind die folgenden:

    - Aus — Die Protokollierung ist für diese Phase nicht aktiviert.
    - Nur Fehler — Die Protokollierung ist nur für Fehler aktiviert.
    - Fehler und Informationsprotokolle — Die Protokollierung ist für alle Ereignisse aktiviert.
    - Vollständige Anfrage- und Antwortprotokolle — Die detaillierte Protokollierung ist für alle Ereignisse aktiviert. Dies kann zur Fehlerbehebung bei APIs hilfreich sein, kann aber dazu führen, dass sensible Daten protokolliert werden.

    > **Note**
    >
    > Es wird empfohlen, Full Request and Response Logs (Vollständige Anforderungs- und Antwortprotokolle) nicht zu verwenden.

11. Wählen Sie Detaillierte Metriken aus, damit API Gateway über CloudWatch die API-Metriken von `API calls` `Latency` `Integration latency`, `400 errors`, und berichtet `500 errors`. Weitere Informationen zu CloudWatch finden Sie in den Abschnitten Grundlegende Überwachung und detaillierte Überwachung im CloudWatch Amazon-Benutzerhandbuch.

    > **Important**
    >
    > Ihr Konto wird für den Zugriff auf Metriken auf Methodenebene belastet, nicht jedoch für CloudWatch Metriken auf API- oder Stufenebene.

12. Aktivieren Sie die Benutzerdefinierte Zugriffsprotokollierung, um die Protokollierung für ein Ziel zuzulassen.

13. Geben Sie für Access Log Destination ARN den ARN einer Protokollgruppe oder eines Firehose ein.

    Das ARN-Format für Firehose ist`arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. Der Name Ihres Firehose-Streams muss sein. `amazon-apigateway-{your-stream-name}`

14. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Weitere Informationen zu Beispielprotokollformaten finden Sie unter the section called “CloudWatch Protokollformate für API Gateway”.

15. Wählen Sie zum Aktivieren der AWS X-Ray-Nachverfolgung für die API-Stufe die Option X-Ray tracing (X-Ray-Nachverfolgung). Weitere Informationen finden Sie unter Ablaufverfolgung von Benutzeranforderungen an REST-APIs mithilfe von X-Ray.

16. Wählen Sie Save Changes (Änderungen speichern). Stellen Sie Ihre API erneut bereit, damit die neuen Einstellungen wirksam werden.

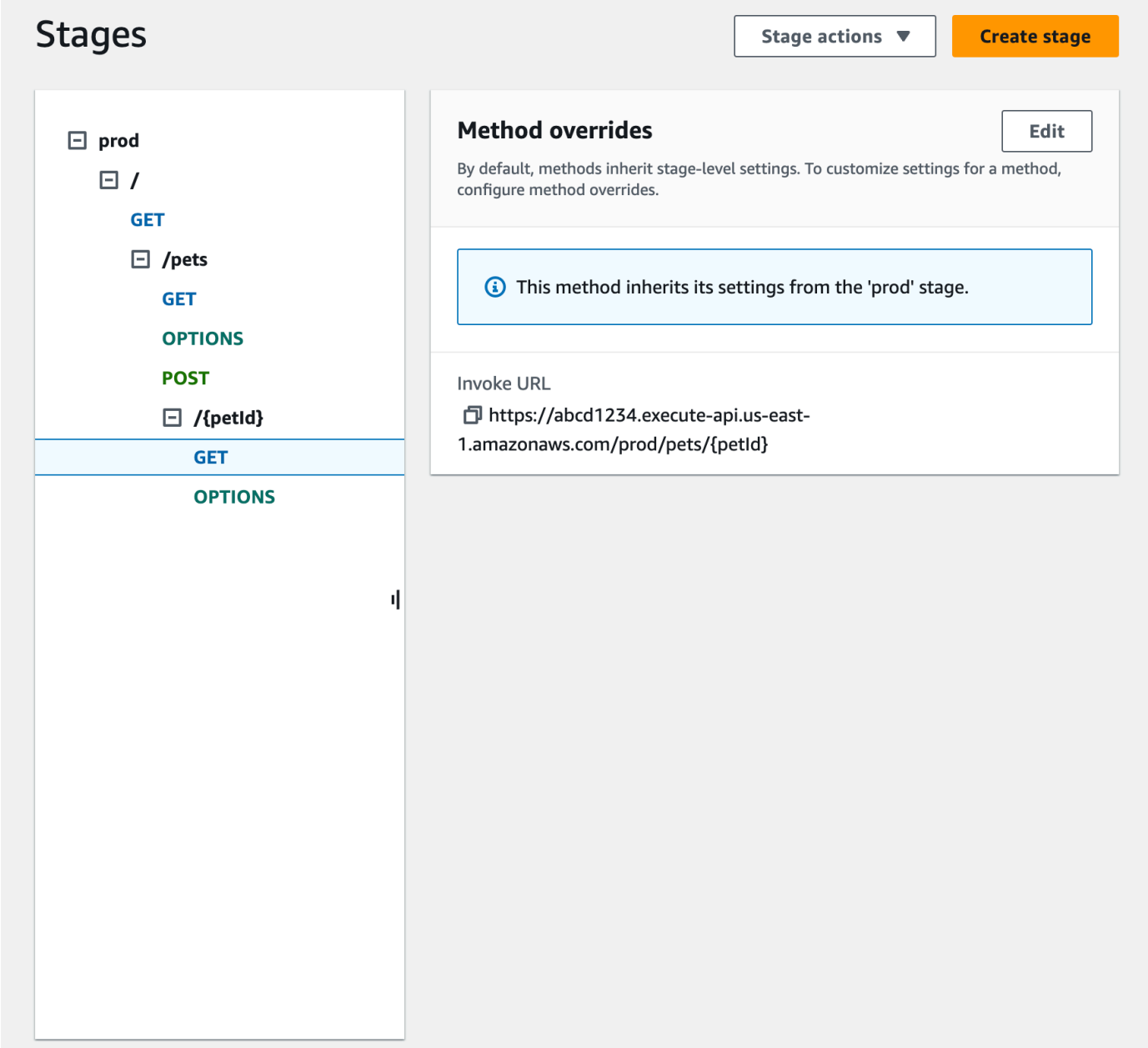
Überschreiben der Einstellungen auf Stufenebene

Die folgenden aktivierten Einstellungen auf Stufenebene können überschrieben werden. Bei einigen dieser Optionen können zusätzliche Gebühren für Sie anfallen. AWS-Konto

Einstellungen auf Stufenebene mithilfe der API Gateway-Konsole überschreiben

So überschreiben Sie Einstellungen auf Stufenebene mithilfe der API Gateway-Konsole

1. Vergrößern Sie die Stufe im sekundären Navigationsbereich, um Methodenüberschreibungen zu konfigurieren und wählen Sie dann eine Methode aus.



2. Klicken Sie für Methodenüberschreibungen auf Bearbeiten.
3. Um CloudWatch Einstellungen auf Methodenebene zu aktivieren, wählen Sie unter CloudWatch Protokolle eine Protokollierungsebene aus.
4. Klicken Sie auf Detaillierte Metriken, um diese auf Methodenebene zu aktivieren. Ihrem Konto wird der Zugriff auf Metriken auf Methodenebene in Rechnung gestellt, nicht jedoch auf CloudWatch Metriken auf API- oder Stufenebene.
5. Klicken Sie auf Drosselung, um diese auf Methodenebene zu aktivieren. Geben Sie die entsprechenden Optionen auf Methodenebene ein. Weitere Informationen zur Drosselung finden Sie unter the section called “Drosselung”.

6. Um den Cache auf Methodenebene zu konfigurieren, wählen Sie Methoden-Cache aktivieren aus. Wenn Sie die Standardeinstellung für das Caching auf Methodenebene in den Stage-Details ändern, hat dies keine Auswirkungen auf diese Einstellung.
7. Wählen Sie Speichern.

Löschen einer Stufe

Wenn Sie eine Stufe nicht mehr benötigen, können Sie diese löschen, um Kosten für ungenutzte Ressourcen zu vermeiden. In den folgenden Schritten wird gezeigt, wie Sie die API Gateway-Konsole zum Löschen einer Stufe verwenden.

> **Warning**
>
> Das Löschen einer Stufe kann dazu führen, dass ein Teil der oder die gesamte zugehörige API nicht mehr von API-Aufrufern verwendet werden kann. Das Löschen einer Stufe kann nicht rückgängig gemacht werden. Sie können die Stufe jedoch neu erstellen und derselben Bereitstellung zuordnen.

Löschen einer Stufe mithilfe der API Gateway-Konsole

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
4. Wählen Sie im Bereich Stages (Stufen) die Stufe aus, die Sie löschen möchten, und klicken Sie dann unter Stage actions (Stufenaktionen) auf Delete stage (Stufe löschen).
5. Geben Sie bei der Aufforderung **`confirm`** ein und klicken Sie dann auf Delete (Löschen).

Einrichten von Tags für eine API-Stufe in API Gateway

In API Gateway können Sie einer API-Stufe ein Tag hinzufügen, das Tag aus der Stufe entfernen oder das Tag anzeigen. Dazu können Sie die API-Gateway-Konsole, das AWS CLI/SDK oder die API-Gateway-REST-API verwenden.

Eine Stufe kann auch Tags von der übergeordneten REST-API erben. Weitere Informationen finden Sie unter the section called “Tag-Vererbung in der Amazon API Gateway V1-API”.

Weitere Informationen über das Markieren mit Tags von API Gateway-Ressourcen finden Sie unter Markieren.

Themen

- Einrichten von Tags für eine API-Stufe mit der API-Gateway-Konsole
- Einrichten von Tags für eine API-Stufe mithilfe der AWS CLI
- Einrichten von Tags für eine API-Stufe mit der API-Gateway-REST-API

Einrichten von Tags für eine API-Stufe mit der API-Gateway-Konsole

Die folgende Vorgehensweise beschreibt, wie Sie Tags für eine API-Stufe einrichten.

So richten Sie Tags für eine API-Stufe mithilfe der API Gateway-Konsole ein

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene API aus oder erstellen Sie eine neue API, die Ressourcen, Methoden und die entsprechenden Integrationen enthält.
3. Wählen Sie eine Stufe aus oder stellen Sie die API in einer neuen Stufe bereit.
4. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
5. Wählen Sie die Registerkarte Tags aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
6. Wählen Sie Tags verwalten aus.
7. Klicken Sie im Tag Editor (Tag-Editor) auf Add Tag (Tag hinzufügen). Geben Sie einen Tag-Schlüssel (z. B. `Department`) in das Feld Schlüssel und einen Tag-Wert (z. B. `Sales`) in das Feld Wert ein. Wählen Sie Save (Speichern) aus, um das Tag zu speichern.
8. Wiederholen Sie bei Bedarf Schritt 5, um der API-Stufe weitere Tags hinzuzufügen. Die maximale Anzahl der Tags pro Stufe beträgt 50.
9. Um ein vorhandenes Tag aus der Stufe zu entfernen, wählen Sie Remove (Entfernen) aus.
10. Wenn die API zuvor in der API Gateway-Konsole bereitgestellt wurde, müssen Sie sie erneut implementieren, damit die Änderungen wirksam werden.

Einrichten von Tags für eine API-Stufe mithilfe der AWS CLI

Sie können Tags für eine API-Stufe mit AWS CLI dem Befehl create-stage oder dem Befehl tag-resource einrichten. Sie können ein oder mehrere Tags aus einer API-Stufe mit dem Befehl untag-resource löschen.

Im folgenden Beispiel wird beim Erstellen einer `test` Stufe ein Tag hinzugefügt:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

Im folgenden Beispiel wird einer `prod` Stufe ein Tag hinzugefügt:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

Im folgenden Beispiel wird das `Department=Sales` Tag aus der `test` Stufe entfernt:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

Einrichten von Tags für eine API-Stufe mit der API-Gateway-REST-API

Sie können Tags für eine API-Stufe mit Hilfe der API Gateway-REST-API einrichten, indem Sie eine der folgenden Aktionen ausführen:

- Rufen Sie `tags:tag` auf, um eine API-Stufe mit einem Tag zu versehen.
- Rufen Sie `tags:untag` auf, um einen oder mehrere Tags aus einer API-Stufe zu löschen.
- Rufen Sie `stage:create` auf, um einen oder mehrere Tags zu einer API-Stufe, die Sie erstellen, hinzuzufügen.

Sie können außerdem `tags:get` aufrufen, um Tags in einer API-Stufe zu beschreiben.

Tag zu einer API-Stufe hinzufügen

Nach Bereitstellung einer API (`m5zr3vnks7`) für eine Stufe (`test`) versehen Sie die Stufe durch Aufruf von `tags:tag` mit einem Tag. Die erforderliche Amazon-Ressourcenname (ARN) (`arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test`) der Stufe muss URL-codiert sein (`arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest`).

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest

{
  "tags" : {
    "Department" : "Sales"
  }
}
```

Sie können auch die vorherige Anforderung verwenden, um einen bestehenden Tag auf einen neuen Wert zu aktualisieren.

Sie können Tags zu einer Stufe hinzufügen, wenn Sie `stage:create` aufrufen, um die Stufe zu erstellen:

```
POST /restapis/<restapi_id>/stages

{
  "stageName" : "test",
  "deploymentId" : "adr134",
  "description" : "test deployment",
  "cacheClusterEnabled" : "true",
  "cacheClusterSize" : "500",
  "variables" : {
    "sv1" : "val1"
  },
  "documentationVersion" : "test",

  "tags" : {
    "Department" : "Sales",
    "Division" : "Retail"
  }
}
```

Tag von einer API-Stufe entfernen

Rufen Sie `Department` auf, um den `tags:untag`-Tag von der Stufe zu entfernen:

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
```

```
Authorization: ...
```

Um mehrere Tags zu entfernen, verwenden Sie eine kommagetrennte Liste mit Tag-Schlüsseln im Abfrageausdruck (zum Beispiel `?tagKeys=Department,Division,…`).

Beschreiben von Tags für eine API-Stufe

Um existierende Tags einer bestehenden Stufe zu beschreiben, rufen Sie auf `tags:get`:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Eine erfolgreiche Antwort ähnelt dem folgenden Beispiel:

```
200 OK

{
    "_links": {
        "curies": {
            "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/
restapi-tags-{rel}.html",
            "name": "tags",
            "templated": true
        },
        "tags:tag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags"
        },
        "tags:untag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags{?tagKeys}",
            "templated": true
        }
    },
    "tags": {
        "Department": "Sales"
    }
}
```

Einrichten von Stufenvariablen für eine REST-API-Bereitstellung

Bei Stufenvariablen handelt es sich um Namen-Wert-Paare, die Sie als Konfigurationsattribute für eine REST-API-Bereitstellungsstufe definieren können. Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung und Mapping-Vorlagen verwendet werden.

Sie können beispielsweise in einer Stufenkonfiguration eine Stufenvariable definieren und dann deren Wert als URL-Zeichenfolge einer HTTP-Integration für eine Methode in Ihrer REST-API festlegen. Später können Sie mithilfe des zugehörigen Stufenvariablennamens (aus der API-Einrichtung) auf die URL-Zeichenfolge verweisen. Auf diese Weise können Sie dieselbe API-Einrichtung mit einem anderen Endpunkt für jede Stufe verwenden, indem Sie den Stufenvariablenwert auf die entsprechenden URLs setzen.

Sie können auch auf Stufenvariablen in Mapping-Vorlagen zugreifen oder Konfigurationsparameter an das AWS Lambda - oder das HTTP-Backend übergeben.

Weitere Informationen zu Mapping-Vorlagen finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen.

> **Note**
>
> Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Genehmigers übergeben. Weitere Informationen hierzu finden Sie unter the section called “Ausgabe von einem API Gateway Lambda Authorizer”.

Anwendungsfälle

Mit den Bereitstellungsstufen in API Gateway können Sie für jede API mehrere Freigabestufen verwalten (z. B. „Alpha“, „Beta“ und „Produktion“). Über die Stufenvariablen können Sie eine API-Bereitstellungsstufe für die Interaktion mit verschiedenen Backend-Endpunkten konfigurieren.

Ihre API kann beispielsweise eine GET-Anforderung als HTTP-Proxy an den Backend-Webhost übergeben (z. B, `http://example.com`). Dabei wird der Backend-Webhost in einer Stufenvariable konfiguriert, so dass API Gateway „example.com“ aufruft, wenn Entwickler Ihren Produktionsendpunkt aufrufen. Wird der Beta-Endpunkt aufgerufen, verwendet API Gateway den in der Stufenvariable für die Beta-Stufe konfigurierten Wert und ruft einen anderen Webhost auf (z. B,

`beta.example.com`). In ähnlicher Weise können Stufenvariablen verwendet werden, um für jede Phase in Ihrer API einen anderen AWS Lambda Funktionsnamen anzugeben.

Sie können auch Stufenvariablen verwenden, um Konfigurationsparameter über Mapping-Vorlagen an eine Lambda-Funktion zu übergeben. Beispielsweise können Sie die gleiche Lambda-Funktion für mehrere Stufen der API nutzen, jedoch soll die Funktion abhängig von der aufgerufenen Stufe jeweils Daten aus einer anderen Amazon DynamoDB-Tabelle lesen. In den Mapping-Vorlagen, von denen die Anforderung für die Lambda-Funktion generiert wird, können Sie den Tabellennamen mit Stufenvariablen an Lambda übergeben.

Beispiele

Damit Sie eine Stufenvariable zur Anpassung des HTTP-Integrationsendpunkts verwenden können, müssen Sie zunächst eine Stufenvariable für einen bestimmten Namen (z. B. **`url`**) konfigurieren und diesem einen Wert zuweisen (z. B. **`example.com`**). Richten Sie als Nächstes in Ihrer Methodenkonfiguration eine HTTP-Proxy-Integration ein. Anstatt die URL des Endpunkts einzugeben, können Sie API Gateway anweisen, den Wert der Stufenvariablen zu verwenden, **`http://${stageVariables.url}`**. Mit diesem Wert wird API Gateway angewiesen, die Stufenvariable `${}` zur Laufzeit abhängig von der ausgeführten API-Stufe zu ersetzen.

Sie können auf ähnliche Weise auf Stufenvariablen verweisen, um einen Lambda-Funktionsnamen, einen AWS Service Proxy-Pfad oder einen AWS Rollen-ARN im Feld mit den Anmeldeinformationen anzugeben.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Wenn Sie eine Lambda-Funktion in der API Gateway Gateway-Konsole angeben, wird ein AWS CLI Befehl angezeigt, mit dem Sie die entsprechenden Berechtigungen konfigurieren können. Sie können dazu auch die AWS Command Line Interface (AWS CLI) verwenden.

```
aws lambda add-permission --function-name "arn:aws:lambda:us-
east-2:123456789012:function:my-function" --source-arn "arn:aws:execute-api:us-
east-2:123456789012:api_id/*/HTTP_METHOD/resource" --principal apigateway.amazonaws.com
 --statement-id apigateway-access --action lambda:InvokeFunction
```

Einrichten von Stufenvariablen mit der Amazon API Gateway-Konsole

In diesem Tutorial erfahren Sie, wie Sie mit der Amazon API Gateway-Konsole Stufenvariablen für zwei Bereitstellungsstufen einer Beispiel-API einrichten. Überprüfen Sie vor Beginn, ob die folgenden Anforderungen erfüllt sind:

- Es muss eine API in API Gateway verfügbar sein. Folgen Sie den Anweisungen in REST-API in API Gateway entwickeln.
- Sie müssen die API mindestens einmal bereitgestellt haben. Folgen Sie den Anweisungen in Bereitstellen einer REST-API in Amazon API Gateway.
- Sie müssen die erste Stufe für eine bereitgestellte API erstellt haben. Folgen Sie den Anweisungen in Erstellen einer neuen Stufe.

So deklarieren Sie Stufenvariablen mit der API Gateway-Konsole

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Erstellen Sie eine API und anschließend eine GET-Methode für die Root-Ressource der API. Stellen Sie den Integrationstyp auf HTTP und die Endpunkt-URL auf **`http://${stageVariables.url}`**.
3. Stellen Sie die API für eine neue Stufe namens **`beta`** bereit.
4. Wählen Sie im Navigationsbereich Stages (Stufen) die Stufe beta.
5. Wählen Sie auf der Registerkarte Stage variables (Stufenvariablen) die Option Edit (Bearbeiten) aus.
6. Wählen Sie Add stage variable (Stufenvariable hinzufügen).
7. Geben Sie unter Name **`url`** ein. Geben Sie für Value (Wert) **`httpbin.org/get`** ein.
8. Wählen Sie Add stage variable (Stufenvariable hinzufügen) und gehen Sie wie folgt vor:

   Geben Sie unter Name **`stageName`** ein. Geben Sie für Value (Wert) **`beta`** ein.
9. Wählen Sie Add stage variable (Stufenvariable hinzufügen) und gehen Sie wie folgt vor:

   Geben Sie unter Name **`function`** ein. Geben Sie für Value (Wert) **`HelloWorld`** ein.

> **Note**
>
> Wenn Sie eine Lambda-Funktion als Wert einer Stufenvariable festlegen, verwenden Sie den lokalen Namen der Funktion und binden den Alias oder die Versionsangabe ein, z. B. **`HelloWorld`**, **`HelloWorld:1`** oder **`HelloWorld:alpha`**. Nutzen Sie nicht den ARN der Funktion (z. B, **`arn:aws:lambda:us-east-1:123456789012:function:HelloWorld`**). Die API Gateway-Konsole

betrachtet den Stufenvariablenwert einer Lambda-Funktion als unvollständigen Funktionsnamen und erweitert die Stufenvariable zu einem ARN.

10. Wählen Sie Speichern.
11. Erstellen Sie nun eine zweite Stufe. Wählen Sie im Navigationsbereich Stages (Stufen) die Option Create stage (Stufe erstellen). Geben Sie für Stage name (Stufenname) **`prod`** ein. Wählen Sie in Deployment (Bereitstellung) eine kürzlich ausgeführte Bereitstellung und anschließend Create stage (Stufe erstellen) aus.
12. Legen Sie wie im Fall der Phase beta die gleichen drei Stufenvariablen (url, stageName und function) auf unterschiedliche Werte fest („**`petstore-demo-endpoint.execute-api.com/petstore/pets`**“, „**`prod`**“ und „**`HelloEveryone`**“).

    Weitere Informationen zur Verwendung von Stufenvariablen finden Sie unter the section called “Verwenden von Stufenvariablen ”.

Verwenden von Amazon API Gateway-Stufenvariablen

Sie können API Gateway-Stufenvariablen verwenden, um auf HTTP- und Lambda-Backends für verschiedene API-Bereitstellungsstufen zuzugreifen. Sie können Stufenvariablen auch verwenden, um stufenbezogene Konfigurationsmetadaten an ein HTTP-Backend als Abfrageparameter und an eine Lambda-Funktion als Nutzlast, die in einer Eingabe-Mapping-Vorlage generiert wurde, zu übergeben.

Voraussetzungen

Sie müssen zwei Stufen mit einer url-Stufenvariable, die auf zwei unterschiedliche HTTP-Endpunkte verweist, erstellen: Die function-Stufenvariable wird zwei verschiedenen Lambda-Funktionen zugewiesen und die stageName-Stufenvariable enthält stufenbezogene Metadaten.

Zugreifen auf einen HTTP-Endpunkt über eine API mit einer Stufenvariable

1. Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hierdurch wird die beta-Stufen-Anforderung `GET` für die Stammressource der API gestartet.

> **Note**
>
> Der Link Invoke URL verweist auf die Stammressource der API auf der beta-Stufe. Durch die Eingabe der URL in einem Webbrowser wird die beta-Stufenmethode `GET` für die Stammressource aufgerufen. Wenn Methoden für untergeordnete Ressourcen und nicht für die Stammressource selbst definiert sind, erhalten Sie bei der Eingabe der URL in einen Webbrowser die Fehlermeldung `{"message":"Missing Authentication Token"}`. In diesem Fall müssen Sie dem Link Invoke URL den Namen einer spezifischen untergeordneten Ressource anfügen.

2. Im Folgenden wird die Antwort angezeigt, die Sie von der beta-Stufen-Anforderung `GET` erhalten. Sie können das Ergebnis auch überprüfen, indem Sie in einem Browser zu http://httpbin.org/get navigieren. Dieser Wert wurde der `url`-Variablen in der beta-Stufe zugewiesen. Die beiden Antworten sind identisch.
3. Wählen Sie im Navigationsbereich Stages die Stufe prod. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hierdurch wird die prod-Stufen-Anforderung `GET` für die Stammressource der API gestartet.
4. Im Folgenden wird die Antwort angezeigt, die Sie von der prod-Stufen-Anforderung `GET` erhalten. Sie können das Ergebnis überprüfen, indem Sie in einem Browser zu http://petstore-demo-endpoint.execute-api.com/petstore/pets navigieren. Dieser Wert wurde der `url`-Variablen in der prod-Stufe zugewiesen. Die beiden Antworten sind identisch.

Übergeben von stufenbezogenen Metadaten an ein HTTP-Backend über eine Stufenvariable in einem Abfrageparameterausdruck

In diesem Verfahren wird beschrieben, wie stufenbezogene Metadaten mittels eines Stufenvariablenwerts in einem Abfrageparameterausdruck an ein HTTP-Backend übergeben werden. Sie verwenden die Stufenvariable `stageName`, die in Einrichten von Stufenvariablen mit der Amazon API Gateway-Konsole deklariert wurde.

1. Wählen Sie im Navigationsbereich Resource die Methode GET.

   Um der URL der Methode einen Abfragezeichenfolge-Parameter hinzuzufügen, wählen Sie die Registerkarte Method Request (Methodenanforderung) und anschließend im Abschnitt Method request settings (Einstellungen der Methodenanforderung) die Option Edit (Bearbeiten).

2. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:

   a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

   b. Geben Sie unter Name **`stageName`** ein.

   c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.

3. Wählen Sie Speichern.

4. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Edit (Bearbeiten) aus.

5. Fügen Sie bei Endpoint URL (Endpunkt-URL) den Wert **`?stageName=${stageVariables.stageName}`** an den zuvor definierten URL-Wert an, damit die gesamte Endpoint URL (Endpunkt-URL) **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`** lautet.

6. Wählen Sie Deploy API (API bereitstellen) und wählen Sie die Stufe beta aus.

7. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen). Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein.

   > **Note**
   >
   > Sie verwenden hier die beta-Stufe, da der HTTP-Endpunkt (von der `url`-Variable mit "http://httpbin.org/get" angegeben) Abfrageparameterausdrücke annimmt und diese in der Antwort als `args`-Objekt zurückgibt.

8. Sie erhalten die folgende Antwort. Beachten Sie, dass `beta` (zugeordnet zur `stageName`-Stufenvariable) als `stageName`-Argument an das Backend übergeben wird.

```
{
  "args": {
    "stageName": "beta"
  },
  "headers": {
    "Accept": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "AmazonAPIGateway_abcd1234",
    "X-Amzn-Apigateway-Api-Id": "abcd1234",
    "X-Amzn-Trace-Id": "Self=1-abcd-11111111111111;Root=1-11111111-111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Aufrufen einer Lambda-Funktion über eine API mit einer Stufenvariable

In diesem Verfahren wird beschrieben, wie Sie eine Stufenvariable verwenden, um eine Lambda-Funktion als Backend der API aufzurufen. Sie verwenden die zuvor deklarierte `function`-Stufenvariable. Weitere Informationen finden Sie unter Einrichten von Stufenvariablen mit der Amazon API Gateway-Konsole.

1. Erstellen Sie eine Lambda-Funktion namens **`HelloWorld`** mit der Laufzeit Node.js. Der Code muss Folgendes enthalten:

   ```
   export const handler = function(event, context, callback) {
       if (event.stageName)
           callback(null, 'Hello, World! I\'m calling from the ' + event.stageName + '
    stage.');
       else
           callback(null, 'Hello, World! I\'m not sure where I\'m calling from...');
   };
   ```

   Weitere Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Erste Schritte mit der REST-API-Konsole.
2. Wählen Sie im Bereich Ressources (Ressourcen) die Option Create resource (Ressource erstellen) aus, und gehen Sie dann wie folgt vor:

   a. Wählen Sie für Resource path (Ressourcenpfad) die Option/aus.

   b. Geben Sie für Resource name (Ressourcenname) **`lambdav1`** ein.

   c. Wählen Sie Create Resource (Ressource erstellen) aus.
3. Wählen Sie die Ressource /lambdav1 und dann Methode erstellen aus.

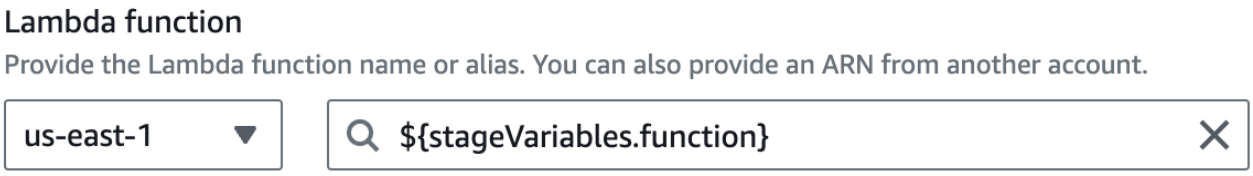
Führen Sie dann die folgenden Schritte aus:

a. Wählen Sie als Methodentyp die Option GET (Abrufen) aus.

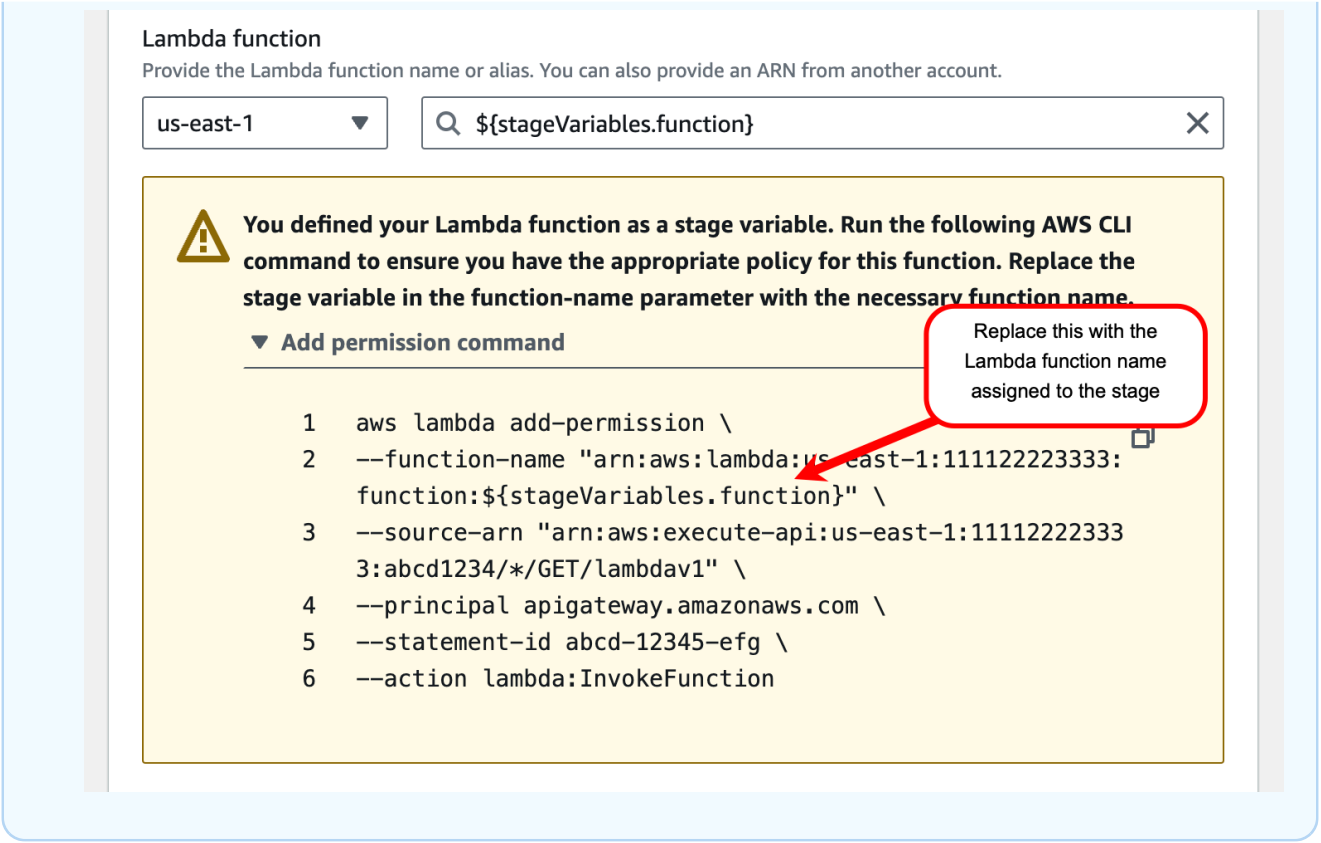
b. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.

c. Lassen Sie Lambda proxy integration (Lambda-Proxyintegration) deaktiviert.

d. Geben Sie für die Lambda function (Lambda-Funktion) `${stageVariables.function}` ein.



> **Tip**
>
> Wenn der Befehl Add permission (Berechtigung hinzufügen) angezeigt wird, kopieren Sie den CLI-Befehl AWS . Sie müssen den Befehl für jede Lambda-Funktion ausführen, die der `function`-Stufenvariable zugewiesen werden soll. Wenn der `$stageVariables.function` Wert beispielsweise lautet, führen Sie den folgenden `HelloWorld` Befehl aus: AWS CLI
>
> ```
> aws lambda add-permission --function-name arn:aws:lambda:us-
> east-1:account-id:function:HelloWorld --source-arn arn:aws:execute-
> api:us-east-1:account-id:api-id/*/GET/lambdav1 --principal
>  apigateway.amazonaws.com --statement-id statement-id-guid --action
>  lambda:InvokeFunction
> ```
>
> Falls das nicht geschieht, führt der Aufruf der Methode zur Fehlermeldung `500 Internal Server Error`. Ersetzen Sie `${stageVariables.function}` durch den Lambda-Funktionsnamen, der dieser Stufenvariable zugeordnet ist.



e. Wählen Sie Methode erstellen aus.

4. Stellen Sie die API sowohl für die Phase **`prod`** als auch für die Phase **`beta`** bereit.
5. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen). Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in einen Webbrowser ein. Hängen Sie **`/lambdav1`** an die URL an, bevor Sie die Eingabetaste drücken.

   Sie erhalten die folgende Antwort.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Übergeben von stufenbezogenen Metadaten an eine Lambda-Funktion über eine Stufenvariable

In diesem Verfahren wird beschrieben, wie stufenbezogene Konfigurationsmetadaten mittels einer Stufenvariable an eine Lambda-Funktion übergeben werden. Erstellen Sie eine POST-Methode und eine Eingabe-Mapping-Vorlage, um Nutzlast mit der zuvor deklarierten `stageName`-Stufenvariable zu generieren.

1. Wählen Sie die Ressource /lambdav1 und dann Methode erstellen aus.

Führen Sie dann die folgenden Schritte aus:

a. Wählen Sie unter Method type (Methodentyp) die Option POST aus.
b. Wählen Sie für den Integration type (Integrationstyp) die Option Lambda function (Lambda-Funktion) aus.
c. Lassen Sie Lambda proxy integration (Lambda-Proxyintegration) deaktiviert.
d. Geben Sie für die Lambda function (Lambda-Funktion) `${stageVariables.function}` ein.
e. Wenn der Befehl Add permission (Berechtigung hinzufügen) angezeigt wird, kopieren Sie den CLI-Befehl AWS . Sie müssen den Befehl für jede Lambda-Funktion ausführen, die der `function`-Stufenvariable zugewiesen werden soll.
f. Wählen Sie Methode erstellen aus.

2. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Edit (Bearbeiten) aus.
3. Wählen Sie Mapping-Vorlagen aus und klicken Sie dann auf Mapping-Vorlage hinzufügen.
4. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
5. Geben Sie für Template body (Vorlagentext) die folgende Vorlage ein:

```
#set($inputRoot = $input.path('$'))
{
    "stageName" : "$stageVariables.stageName"
}
```

> **Note**
>
> In einer Mapping-Vorlage muss eine Stufenvariable in Anführungszeichen (wie in `"$stageVariables.stageName"` oder `"${stageVariables.stageName}"`) referenziert werden. An anderen Stellen muss sie ohne Anführungszeichen (wie in) referenziert werd `${stageVariables.function}`).

6. Wählen Sie Speichern.
7. Stellen Sie die API sowohl für die Phase **`beta`** als auch für die Phase **`prod`** bereit.
8. Gehen Sie wie folgt vor, um einen REST-API-Client zum Übergeben von stufenspezifischen Metadaten zu verwenden:

a. Wählen Sie im Navigationsbereich Stages beta. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in das Eingabefeld eines REST-API-Clients ein. Hängen Sie **`/lambdav1`** an, bevor Sie Ihre Anfrage absenden.

   Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the beta stage."
```

b. Wählen Sie im Navigationsbereich Stages (Stufen) die Option prod. Wählen Sie unter Stage details (Stufendetails) das Kopiersymbol, um die Aufruf-URL Ihrer API zu kopieren, und geben Sie dann die Aufruf-URL Ihrer API in das Eingabefeld eines REST-API-Clients ein. Hängen Sie **`/lambdav1`** an, bevor Sie Ihre Anfrage absenden.

   Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the prod stage."
```

9. Gehen Sie wie folgt vor, um die Funktion Test zum Übergeben von stufenspezifischen Metadaten zu verwenden:

   a. Wählen Sie im Navigationsbereich Resources (Ressourcen) die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.

   b. Geben Sie für function **`HelloWorld`** ein.

   c. Geben Sie für stageName **`beta`** ein.

   d. Wählen Sie Test aus. Sie müssen Ihrer `POST`-Anfrage keinen Text hinzufügen.

      Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the beta stage."
```

   e. Sie können die vorherigen Schritte wiederholen, um die Prod-Stufe zu testen. Geben Sie für stageName **`Prod`** ein.

      Sie erhalten die folgende Antwort.

```
"Hello, World! I'm calling from the prod stage."
```

Referenz für Amazon API Gateway-Stufenvariablen

Sie können die API Gateway-Stufenvariablen in folgenden Fällen verwenden.

Parameter-Mapping-Ausdrücke

Eine Stufenvariable kann in einem Parameter-Mapping-Ausdruck für den Header-Parameter einer API-Methodenanforderung oder -antwort genutzt werden, und zwar ohne Teilsubstitution. Im folgenden Beispiel wird die Stufenvariable ohne $ und umschließende `{...}` referenziert.

- `stageVariables.<variable_name>`

Mapping-Vorlagen

Eine Stufenvariable kann überall in einer Mapping-Vorlage verwendet werden, wie in den folgenden Beispielen dargestellt.

- `{ "name" : "$stageVariables.<variable_name>"}`
- `{ "name" : "${stageVariables.<variable_name>}"}`

HTTP-Integrations-URIs

Eine Stufenvariable kann als Teil einer HTTP-Integrations-URL verwendet werden, wie in den folgenden Beispielen dargestellt:

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS Integrations-URIs

Eine Stufenvariable kann als Teil von AWS URI-Aktions- oder Pfadkomponenten verwendet werden, wie im folgenden Beispiel gezeigt.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS Integrations-URIs (Lambda-Funktionen)

Eine Stufenvariable kann anstelle des Lambda-Funktionsnamens (oder Version/Alias) verwendet werden, wie in den folgenden Beispielen dargestellt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

Amazon-Cognito-Benutzerpool

Eine Stufenvariable kann anstelle eines Amazon Cognito Cognito-Benutzerpools für einen `COGNITO_USER_POOLS` Autorisierer verwendet werden.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWS Anmeldeinformationen für die Integration

Eine Stufenvariable kann als Teil des ARN für AWS Benutzer-/Rollenanmeldedaten verwendet werden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Einrichten einer API Gateway-Canary-Release-Bereitstellung

Canary-Release ist eine Softwareentwicklungsstrategie, bei der eine neue Version einer API (und anderer Software) zu Testzwecken eingesetzt wird und die Basisversion weiterhin als Produktionsrelease für den normalen Betrieb auf derselben Stufe eingesetzt wird. Im Rahmen dieses Artikels gehen wir davon aus, dass die Basisversion das Produktionsrelease ist. Sie können das Canary-Release jedoch auch für Tests in jeder anderen Version verwenden.

Bei einer Canary-Release-Bereitstellung wird der gesamte API-Verkehr mit einem vorkonfigurierten Verhältnis nach dem Zufallsprinzip in ein Produktionsrelease und ein Canary-Release getrennt. Normalerweise erhält die Canary-Release einen kleinen Prozentsatz des API-Verkehrs und das Produktionsrelease nimmt den Rest in Anspruch. Die aktualisierten API-Features sind nur für den API-Verkehr über das Canary-Release sichtbar. Sie können den Prozentsatz des Canary-Datenverkehrs anpassen, um die Testabdeckung oder die Leistung zu optimieren.

Indem man den Canary-Verkehr gering und die Auswahl zufällig hält, werden die meisten Benutzer zu keiner Zeit durch potenzielle Fehler in der neuen Version beeinträchtigt und kein einzelner Benutzer wird ständig beeinträchtigt.

Nachdem die Testmetriken Ihre Anforderungen erfüllt haben, können Sie das Canary-Release in das Produktionsrelease überführen und Canary für die Bereitstellung deaktivieren. Damit stehen die neuen Features bereits in der Produktionsstufe zur Verfügung.

Themen

Canary-Release-Bereitstellung in API Gateway

In API Gateway verwendet eine Canary-Release-Bereitstellung die Bereitstellungsstufe für die Produktionsfreigabe der Basisversion einer API und fügt der Stufe ein Canary-Release für die neuen Versionen relativ zur Basisversion der API hinzu. Die Stufe ist der ersten Bereitstellung und das Canary-Release den nachfolgenden Bereitstellungen zugeordnet. Zu Beginn verweisen sowohl die Stufe als auch das Canary auf die gleiche API-Version. Wir nutzen die Begriffe Stufen- und Produktionsrelease und Canary und Canary-Release in diesem Abschnitt gleichwertig.

Um eine API mit einem Canary-Release bereitzustellen, erstellen Sie eine Canary-Release-Bereitstellung, indem Sie die Canary-Einstellungen zur Stufe einer regulären Bereitstellung hinzufügen. Die Canary-Einstellungen beschreiben das zugrunde liegende Canary-Release und die Stufe repräsentiert die Produktionsfreigabe der API innerhalb dieser Bereitstellung. Um Canary-Einstellungen hinzuzufügen, legen Sie `canarySettings` in der Bereitstellungsstufe fest und geben Sie Folgendes an:

- Eine Bereitstellungs-ID. Diese ID ist zunächst mit der ID der Basisversion, die für die Stufe festgelegt wurde, identisch.
- Ein Prozentsatz des API-Verkehrs zwischen 0,0 und 100,0 (inklusive) für das Canary-Release.
- Stufenvariablen für das Canary-Release, die die Stufenvariablen des Produktionsreleases überschreiben können.
- Die Verwendung des Stufen-Caches für Canary-Anforderungen, wenn useStageCache festgelegt ist und das API-Caching für die Stufe aktiviert ist.

Nachdem ein Canary-Release aktiviert wurde, kann die Bereitstellungsstufe nicht mit einem anderen nicht-Canary-Release verbunden werden, bis das Canary-Release deaktiviert und die Canary-Einstellungen von der Stufe entfernt wurden.

Wenn Sie die Protokollierung der API-Ausführung aktivieren, werden für das Canary-Release eigenen Protokolle und Metriken für alle Canary-Anforderungen generiert. Sie werden an eine CloudWatch Logs-Protokollgruppe der Produktionsstufe sowie an eine Canary-spezifische CloudWatch Logs-Protokollgruppe gemeldet. Gleiches gilt für die Zugriffsprotokollierung. Die separaten Canary-spezifischen Protokolle sind hilfreich, um neue API-Änderungen zu validieren und zu entscheiden, ob die Änderungen akzeptiert werden sollen und das Canary-Release in die Produktionsstufe übernommen werden soll oder ob die Änderungen verworfen werden sollen und das Canary-Release aus der Produktionsstufe zurückgenommen werden soll.

Die Protokollgruppe für die Ausführung in der Produktionsstufe heißt `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` und die Protokollgruppe für die Ausführung im Canary-Release heißt `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Für die Zugriffsprotokollierung müssen Sie eine neue Protokollgruppe anlegen oder eine bestehende auswählen. Der Name der Canary-Release-Zugriffsprotokollgruppe enthält das Suffix `/Canary`, das an den Namen der ausgewählten Protokollgruppe angehängt ist.

Ein Canary-Release kann den Stage-Cache nutzen (wenn er aktiviert ist), um Antworten zu speichern und Cache-Einträge zu verwenden, um die Ergebnisse innerhalb einer vorkonfigurierten Time-to-Live-Periode (TTL) an die nächsten Canary-Anforderungen zurückzugeben.

Bei einer Canary-Release-Bereitstellung können das Produktions- und das Canary-Release der API der gleichen Version oder verschiedenen Versionen zugeordnet werden. Wenn sie verschiedenen Versionen zugeordnet sind, werden die Antworten für Produktions- und Canary-Anforderungen getrennt zwischengespeichert und der Stage-Cache liefert entsprechende Ergebnisse für Produktions- und Canary-Anforderungen. Wenn das Produktionsrelease und das Canary-Release mit derselben Bereitstellung verbunden sind, verwendet der Stage-Cache einen einzigen Cache-Key für beide Arten von Anforderungen und gibt dieselbe Antwort für dieselben Anforderungen aus dem Produktionsrelease und dem Canary-Release zurück.

Erstellen einer Canary-Release-Bereitstellung

Sie erstellen eine Canary-Release-Bereitstellung bei der Bereitstellung der API mit Canary-Einstellungen als zusätzliche Eingabe für die Bereitstellungserstellung.

Sie können auch aus einer bestehenden Nicht-Canary-Bereitstellung eine Canary-Release-Bereitstellung erstellen, indem Sie eine `stage:update`-Anforderung erstellen, um die Canary-Einstellungen für die Stufe hinzuzufügen.

Beim Erstellen einer nicht-Canary-Release-Bereitstellung können Sie einen nicht existierenden Stufennamen angeben. API Gateway erstellt eine Stufe, wenn die angegebene Stufe nicht existiert. Sie können jedoch beim Anlegen einer Canary-Release-Bereitstellung keinen nicht existierenden Stufennamen angeben. Sie erhalten eine Fehlermeldung und API Gateway erstellt keine Canary-Release-Bereitstellung.

Sie können eine Canary-Release-Bereitstellung in API Gateway mit der API-Gateway-Konsole, der AWS CLI oder einem AWS SDK erstellen.

Themen

- Erstellen einer Canary-Bereitstellung mithilfe der API Gateway-Konsole
- Erstellen einer Canary-Release-Bereitstellung mit der AWS CLI

Erstellen einer Canary-Bereitstellung mithilfe der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole zu erstellen:

So erstellen Sie eine erste Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene REST API aus oder erstellen Sie eine neue REST API.
3. Klicken Sie im Hauptnavigationsbereich von auf Resources (Ressourcen) und auf Deploy API (API bereitstellen). Befolgen Sie die Anweisungen auf der Seite in Deploy API, um die API in einer neuen Stufe bereitzustellen.

   Bislang haben Sie die API in einer Produktions-Release-Stufe bereitgestellt. Als Nächstes konfigurieren Sie Canary-Einstellungen für die Stufe und, falls erforderlich, auch das Caching. Sie legen Stufenvariablen fest oder konfigurieren die API-Ausführung oder die Zugriffsprotokolle.
4. Um das API-Caching zu aktivieren oder der Stufe eine AWS WAF Web-ACL zuzuordnen, wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus. Weitere Informationen finden Sie unter the section called “Cache-Einstellungen” oder the section called “So verknüpfen Sie mithilfe der API Gateway Gateway-Konsole eine AWS WAF Web-ACL mit einer API-Gateway-API-Stufe”.
5. Um die Ausführung zu konfigurieren oder auf die Protokollierung zuzugreifen, wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) und folgen Sie den Anweisungen am Bildschirm. Weitere Informationen finden Sie unter CloudWatch Protokollierung für eine REST-API in API Gateway einrichten.
6. Um Stufenvariablen festzulegen, wählen Sie die Registerkarte Stage variables (Stufenvariablen) und folgen Sie den Anweisungen am Bildschirm, um Stufenvariablen hinzuzufügen oder zu ändern. Weitere Informationen finden Sie unter the section called “Einrichten von Stufenvariablen”.
7. Wählen Sie die Registerkarte Canary und dann Create canary (Canary erstellen) aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Canary anzuzeigen.
8. Geben Sie unter Canary settings (Canary-Einstellungen) für Canary den Prozentsatz der Anfragen ein, die an den Canary weitergeleitet werden sollen.
9. Wählen Sie bei Bedarf Stage Cache (Stufen-Cache) aus, um das Caching für die Canary-Version zu aktivieren. Der Cache steht für das Canary-Release erst dann zur Verfügung, wenn das API-Caching aktiviert ist.
10. Um bestehende Stufenvariablen zu überschreiben, geben Sie für Canary Override (Überschreibung des Canary) einen neuen Wert für die Stufenvariable ein.

Nachdem das Canary-Release in der Bereitstellungsstufe initialisiert wurde, ändern Sie die API und testen die Änderungen. Sie können die API in derselben Stufe erneut bereitstellen, sodass sowohl die aktualisierte Version als auch die Basisversion über dieselbe Stufe zugreifbar sind. In den folgenden Schritten wird beschrieben, wie Sie dies tun.

So stellen Sie die neueste API-Version für ein Canary bereit

1. Wählen Sie bei jeder Aktualisierung der API Deploy API (API bereitstellen) aus.
2. Wählen Sie in Deploy API (API bereitstellen) in der Dropdown-Liste Deployment stage (Bereitstellungsstufe) die Stufe aus, die einen Canary enthält.
3. (Optional) Geben Sie unter Deployment description (Beschreibung der Bereitstellen) eine Beschreibung ein.
4. Wählen Sie Deploy, um die neueste API-Version auf das Canary-Release zu übertragen.
5. Falls erwünscht, rekonfigurieren Sie die Stufeneinstellungen, Protokolle oder Canary-Einstellungen, wie in beschriebe So erstellen Sie eine erste Canary-Release-Bereitstellung.

Dadurch verweist das Canary-Release auf die neueste Version, während das Produktionsrelease noch auf die erste Version der API verweist. canarySettings hat jetzt einen neuen deploymentId-Wert, während die Stufe noch den ursprünglichen deploymentId-Wert aufweist. Hinter den Kulissen ruft die Konsole stage:update auf.

Erstellen einer Canary-Release-Bereitstellung mit der AWS CLI

Erstellen Sie zunächst eine Basisbereitstellung mit zwei Stufenvariablen, jedoch ohne Canary:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'prod' \
```

Der Befehl gibt eine Darstellung der resultierenden `Deployment` zurück, ähnlich der folgenden:

```
{
    "id": "du4ot1",
    "createdDate": 1511379050
}
```

Die resultierende Bereitstellungs-`id` identifiziert einen Snapshot (oder eine Version) der API.

Erstellen Sie jetzt eine Canary-Bereitstellung in der `prod`-Stufe:

```
aws apigateway create-deployment  --rest-api-id abcd1234 \
    --canary-settings \
    '{
        "percentTraffic":10.5,
        "useStageCache":false,
        "stageVariableOverrides":{
            "sv1":"val2",
            "sv2":"val3"
        }
    }' \
    --stage-name 'prod'
```

Wenn die angegebene Stufe (`prod`) nicht existiert, gibt der vorhergehende Befehl einen Fehler zurück. Andernfalls wird die neu erstellte deployment-Ressourcenrepräsentation ähnlich der folgenden zurückgegeben:

```
{
    "id": "a6rox0",
    "createdDate": 1511379433
}
```

Die resultierende Bereitstellungs-`id` identifiziert die Testversion der API für das Canary-Release. Dadurch ist die zugehörige Stufe für Canary aktiviert. Sie können sich diese Stufenrepräsentation ansehen, indem Sie den Befehl `get-stage` aufrufen, ähnlich wie im Folgenden beschrieben:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Nachfolgend ist eine Repräsentation der `Stage` als Ausgabe des Befehls zu sehen:

```
{
    "stageName": "prod",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "du4ot1",
```

```
    "lastUpdatedDate": 1511379433,
    "createdDate": 1511379050,
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "a6rox0",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}
```

In diesem Beispiel verwendet die Basisversion der API die Stufenvariablen `{"sv0":val0", "sv1":val1"}`, während die Testversion die Stufenvariablen `{"sv1":val2", "sv2":val3"}` verwendet. Sowohl das Produktionsrelease als auch das Canary-Release verwenden die gleiche Stufenvariable `sv1`, jedoch mit unterschiedlichen Werten (`val1` und `val2`). Die Stufenvariable `sv0` wird ausschließlich im Produktionsrelease verwendet und die Stufenvariable `sv2` wird nur im Canary-Release verwendet.

Sie können aus einer bestehenden, regulären Bereitstellung eine Canary-Release-Bereitstellung erstellen, indem Sie die Stufe aktualisieren, um Canary zu aktivieren. Um dies zu demonstrieren, erstellen Sie zunächst eine reguläre Bereitstellung:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'beta'
```

Der Befehl gibt eine Repräsentation der Basisversionsbereitstellung zurück:

```
{
    "id": "cifeiw",
    "createdDate": 1511380879
}
```

Die dazugehörige Beta-Stufe hat keine Canary-Einstellungen:

```
{
```

```
    "stageName": "beta",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "cifeiw",
    "lastUpdatedDate": 1511380879,
    "createdDate": 1511380879,
    "methodSettings": {}
}
```

Erstellen Sie jetzt eine neue Canary-Release-Bereitstellung, indem Sie ein Canary zur Stufe hinzufügen:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name 'beta' \
    --patch-operations '[{
            "op": "replace",
            "value": "0.0",
            "path": "/canarySettings/percentTraffic"
        }, {
            "op": "copy",
            "from": "/canarySettings/stageVariableOverrides",
            "path": "/variables"
        }, {
            "op": "copy",
            "from": "/canarySettings/deploymentId",
            "path": "/deploymentId"
        }]'
```

Eine Repräsentation der aktualisierten Stufe sieht so aus:

```
{
    "stageName": "beta",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
```

```
    "deploymentId": "cifeiw",
    "lastUpdatedDate": 1511381930,
    "createdDate": 1511380879,
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "cifeiw",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}
```

Da wir gerade ein Canary für eine existierende Version der API aktiviert haben, zeigen sowohl das Produktions-Release (`Stage`) als auch das Canary-Release (`canarySettings`) auf die gleiche Bereitstellung – d. h. die gleiche Version (`deploymentId`) der API. Nachdem Sie die API geändert und für diese Stufe wieder bereitgestellt haben, ist die neue Version im Canary-Release. Die Basisversion verbleibt im Produktions-Release. Dies wirkt sich auf die Stufenentwicklung aus, wenn die `deploymentId` im Canary-Release auf die neue Bereitstellungs-`id` aktualisiert wird und die `deploymentId` im Produktions-Release unverändert bleibt.

Aktualisieren eines Canary-Releases

Nach der Bereitstellung eines Canary-Releases können Sie den prozentualen Anteil des Canary-Verkehrs anpassen oder die Verwendung eines Stage-Caches aktivieren oder deaktivieren, um die Testperformance zu optimieren. Sie können außerdem Stufenvariablen modifizieren, die im Canary-Release verwendet werden, wenn der Ausführungskontext aktualisiert wird. Um solche Aktualisierungen vorzunehmen, rufen Sie die Operation stage:update mit neuen Werten für canarySettings auf.

Sie können eine Canary-Release mithilfe der API-Gateway-Konsole, dem AWS CLI-Befehl update-stage oder einem AWS SDK aktualisieren.

Themen

- Aktualisieren eines Canary-Releases mit der API Gateway-Konsole
- Aktualisieren eines Canary-Releases mit der AWS CLI

Aktualisieren eines Canary-Releases mit der API Gateway-Konsole

Um die API Gateway-Konsole zur Aktualisierung der vorhandenen Canary-Einstellungen einer Stufe zu nutzen, führen Sie die folgenden Schritte aus:

Aktualisieren bestehender Canary-Einstellungen

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende REST API aus.
2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary und anschließend Edit (Bearbeiten) aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Canary anzuzeigen.
4. Aktualisieren Sie Request distribution (Verteilung anfordern), indem Sie den Prozentsatz auf einen Wert zwischen 0,0 und 100,0 (inklusive) erhöhen oder verringern.
5. Aktivieren oder deaktivieren Sie das Kontrollkästchen unter Stage cache (Stage-Cache).
6. Fügen Sie Canary stage variables (Canary-Stufenvariablen) hinzu, entfernen oder ändern Sie sie.
7. Wählen Sie Speichern.

Aktualisieren eines Canary-Releases mit der AWS CLI

Um mit der AWS CLI ein Canary zu aktualisieren, rufen Sie den Befehl `update-stage` auf.

Um die Verwendung eines Stufen-Caches für das Canary zu aktivieren oder zu deaktivieren, rufen Sie den Befehl `update-stage` wie folgt auf:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Um den Prozentsatz des Canary-Verkehrs anzupassen, rufen Sie `update-stage` auf, um den Wert `/canarySettings/percentTraffic` für die Stufe zu ersetzen.

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
```

```
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Um Canary-Stufenvariablen zu aktualisieren, einschließlich Hinzufügen, Ersetzen oder Entfernen von Canary-Stufenvariablen:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
      }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/var2",
        "value": "val4"
      }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
      }]'
```

Sie können das oben Genannte aktualisieren, indem Sie die Operationen zu einem einzigen `patch-operations`-Wert zusammenfassen:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/percentTraffic",
        "value": "20.0"
    }, {
        "op": "replace",
        "path": "/canarySettings/useStageCache",
        "value": "true"
    }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
```

```
        "value": "newVal"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/val2",
        "value": "val4"
      }]'
```

Hochstufen eines Canary-Releases

Um ein Canary-Release hochzustufen, machen Sie es in der Produktionsstufe der zu testenden API-Version verfügbar. Der Vorgang umfasst die folgenden Aufgaben:

- Setzen Sie die Bereitstellungs-ID der Stufe mit den Bereitstellungs-ID-Einstellungen des Canarys zurück. Dadurch wird der API-Snapshot der Stufe mit dem Snapshot des Canarys aktualisiert, sodass die Testversion auch zum Produktionsrelease wird.
- Aktualisieren Sie Stufenvariablen mit Canary-Stufenvariablen, falls vorhanden. Dadurch wird der API-Ausführungskontext der Stufe mit dem des Canarys aktualisiert. Ohne dieses Update kann die neue API-Version unerwartete Ergebnisse liefern, wenn die Testversion unterschiedliche Stufenvariablen oder unterschiedliche Werte vorhandener Stufenvariablen verwendet.
- Legen Sie den Prozentsatz des Canary-Verkehrs auf 0,0 % fest.

Das Hochstufen eines Canary-Releases bedeutet nicht, dass das Canary für die Stufe deaktiviert wird. Um ein Canary zu deaktivieren, müssen Sie die Canary-Einstellungen der Stufe entfernen.

Themen

- Hochstufen eines Canary-Releases mit der API Gateway-Konsole
- Hochstufen eines Canary-Releases mit der AWS CLI

Hochstufen eines Canary-Releases mit der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole hochzustufen:

Hochstufen einer Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende API im Hauptnavigationsbereich aus.

2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary.
4. Wählen Sie Promote Canary (Canary hochstufen) aus.
5. Bestätigen Sie die durchzuführenden Änderungen und wählen Sie Promote canary (Canary hochstufen) aus.

Nach der Aktion verweist das Produktionsrelease auf die gleiche API-Version (deploymentId) wie das Canary-Release. Sie können dies mithilfe der überprüfe AWS CLI. Für Beispiele vgl. the section called “Hochstufen eines Canary-Releases mit der AWS CLI”.

Hochstufen eines Canary-Releases mit der AWS CLI

Um mit den AWS CLI-Befehlen ein Canary-Release für das Produktionsrelease hochzustufen, rufen Sie den Befehl `update-stage` auf, um die dem Canary zugeordnete `deploymentId` zu der der Stufe zugeordneten `deploymentId` zu kopieren, den Prozentsatz für den Canary-Verkehr auf null (`0.0`) zurückzusetzen und alle Canary-Stufenvariablen zur entsprechenden Stufe zu kopieren.

Angenommen, wir haben eine Canary-Release-Bereitstellung, die durch eine in etwa wie folgt aussehende Stufe beschrieben wird:


```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 10.5
    },
    "createdDate": "2017-11-20T04:42:19Z",
```

```
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": "2017-11-22T00:54:28Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
    "variables": {
        "sv1": "val1"
    }
}
```

Wir rufen die folgende `update-stage`-Anforderung zum Hochstufen auf:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id}  \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
        "value": "0.0",
        "path": "/canarySettings/percentTraffic"
      }, {
        "op": "copy",
        "from": "/canarySettings/stageVariableOverrides",
        "path": "/variables"
      }, {
        "op": "copy",
        "from": "/canarySettings/deploymentId",
        "path": "/deploymentId"
      }]'
```

Nach der Werbeaktion sieht die Stufe nun folgendermaßen aus:

```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
```

```
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 0
    },
    "createdDate": "2017-11-20T04:42:19Z",
    "deploymentId": "eh1sby",
    "lastUpdatedDate": "2017-11-22T05:29:47Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
    "variables": {
        "sv2": "val3",
        "sv1": "val2"
    }
}
```

Wie Sie sehen, wird beim Heraufstufen einer Canary-Release auf die Stufe die Canary nicht deaktivieren. Die Bereitstellung ist weiterhin eine Canary-Release-Bereitstellung. Um daraus eine reguläre Produktions-Release-Bereitstellung zu machen, müssen Sie die Canary-Einstellungen deaktivieren. Weitere Informationen dazu, wie Sie eine Canary-Release-Bereitstellung deaktivieren, finden Sie unter the section called “Deaktivieren eines Canary-Releases”.

Deaktivieren eines Canary-Releases

Um eine Canary-Release-Bereitstellung zu deaktivieren, müssen Sie `canarySettings` auf null setzen, um es von der Stufe zu entfernen.

Sie können eine Canary-Release-Bereitstellung mit der API-Gateway-Konsole, der AWS CLI oder einem AWS SDK deaktivieren.

Themen

- Deaktivieren eines Canary-Releases mit der API Gateway-Konsole
- Deaktivieren eines Canary-Releases mit der AWS CLI

Deaktivieren eines Canary-Releases mit der API Gateway-Konsole

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit der API Gateway-Konsole zu deaktivieren:

Deaktivieren einer Canary-Release-Bereitstellung

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie eine bestehende API im Hauptnavigationsbereich aus.
2. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine bestehende Stufe aus.
3. Wählen Sie die Registerkarte Canary.
4. Wählen Sie Löschen.
5. Bestätigen Sie, dass Sie das Canary löschen möchten, indem Sie Delete auswählen.

Dadurch wird die Eigenschaft `canarySettings` auf `null` gesetzt und aus der Bereitstellungsstufe entfernt. Sie können dies mithilfe der überprüfe AWS CLI. Für Beispiele vgl. the section called “Deaktivieren eines Canary-Releases mit der AWS CLI”.

Deaktivieren eines Canary-Releases mit der AWS CLI

Gehen Sie nach den Anweisungen unten vor, um eine Canary-Release-Bereitstellung mit dem AWS CLI-`update-stage`-Befehl zu deaktivieren:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name canary \
    --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

Eine erfolgreiche Antwort gibt eine Nutzlast ähnlich der folgenden zurück:

```
{
    "stageName": "prod",
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "nfcn0x",
```

```
    "lastUpdatedDate": 1511309280,
    "createdDate": 1511152939,
    "methodSettings": {
        ...
    }
}
```

Wie aus der Ausgabe hervorgeht, ist die Eigenschaft `canarySettings` nun nicht mehr in der Stufe einer Canary-deaktivierten Bereitstellung enthalten.

Aktualisierungen für eine REST-API, die eine erneute Bereitstellung erfordern

Die Pflege einer API ist gleichzusetzen mit der Anzeige, Aktualisierung und Löschung der vorhandenen API-Einrichtungen. Sie können eine API mithilfe der API Gateway Gateway-Konsole AWS CLI, eines SDK oder der API Gateway Gateway-REST-API verwalten. Die Aktualisierung einer API umfasst die Änderung bestimmter Ressourceneigenschaften oder Konfigurationseinstellungen der API. Ressourcenaktualisierungen erfordern die erneute Bereitstellung der API, Konfigurationsaktualisierungen jedoch nicht.

API-Ressourcen, die aktualisiert werden können, finden Sie in der folgenden Tabelle.

API-Ressourcenaktualisierungen, die die erneute Bereitstellung der API erfordern

Ressource	Anmerkungen
ApiKey	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter apikey:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
Authorizer	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter authorizer:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
DocumentationPart	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter documentationpart:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
DocumentationVersion	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter documentationversion:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.

Ressource	Anmerkungen
GatewayResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter gatewayresponse:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
Integration	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter integration:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
IntegrationResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter integrationresponse:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
Methode	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter method:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
MethodResponse	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter methodresponse:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
Model	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter model:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
RequestValidator	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter requestvalidator:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
Ressource	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter resource:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.
RestApi	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter restapi:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.

Ressource	Anmerkungen
VpcLink	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter vpclink:update. Die Aktualisierung erfordert die erneute Bereitstellung der API.

API-Konfigurationen, die aktualisiert werden können, finden Sie in der folgenden Tabelle.

API-Konfigurationsaktualisierungen, die keine erneute Bereitstellung der API erfordern

Konfiguration	Anmerkungen
Account	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter account:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Bereitstellung	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter deployment:update.
DomainName	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter domainname:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.
BasePathMapping	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter basepathmapping:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Stage	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter stage:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.
Usage	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter usage:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.
UsagePlan	Informationen zu den entsprechenden Eigenschaften und unterstützten Operationen finden Sie unter usageplan:update. Die Aktualisierung erfordert keine erneute Bereitstellung der API.

Einrichten von benutzerdefinierten Domänennamen für REST-APIs

Benutzerdefinierte Domänennamen sind einfachere und intuitivere URLs, die Sie Ihren API-Benutzern zur Verfügung stellen können.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

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

wo von API Gateway generiert `api-id` wird, `region` (AWS Region) wird von Ihnen bei der Erstellung der API angegeben und `stage` wird von Ihnen bei der Bereitstellung der API angegeben.

Der Hostname-Teil der URL (d. h. *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`) verweist auf einen API-Endpunkt. Der Standard-API-Endpunkt hat möglicherweise einen schwer zu merkenden, wenig benutzerfreundlichen Namen.

Mit benutzerdefinierten Domänennamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. `myservice`) auswählen, um die alternative URL Ihrer API zuzuordnen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

> **Note**
>
> Eine regionale benutzerdefinierte Domäne kann REST-APIs and HTTP-APIs zugeordnet werden. Sie können API Gateway Version 2-APIs verwenden, um regionale benutzerdefinierte Domänennamen für REST-APIs zu erstellen und zu verwalten.
> Benutzerdefinierte Domänennamen werden für private APIs nicht unterstützt.
> Sie können eine von Ihrer REST-API unterstützte minimale TLS-Version auswählen. Für REST-APIs können Sie TLS 1.2 oder TLS 1.0 auswählen.

Registrieren eines Domänennamens

Wenn Sie benutzerdefinierte Domänennamen für Ihre APIs einrichten möchten, müssen Sie eine Internetdomäne registrieren. Ihr Domainname muss der RFC 1035-Spezifikation entsprechen und darf ein Maximum von 63 kn pro Label und 255 kn insgesamt haben. Falls erforderlich, können Sie

eine Internetdomäne mit Amazon Route 53 oder mit einem Domain-Registrar Ihrer Wahl registrieren. Der benutzerdefinierte Domänenname einer API kann dem Namen einer Unter- oder Stammdomäne (auch als „Zone Apex“ bezeichnet) einer registrierten Internetdomäne entsprechen.

Nachdem ein benutzerdefinierter Domänenname in API Gateway erstellt wurde, müssen Sie den Ressourceneintrag Ihres DNS-Providers erstellen oder aktualisieren, um ihn Ihrem API-Endpunkt zuzuordnen. Ohne ein solches Mapping können API-Anfragen für den benutzerdefinierten Domänennamen API Gateway nicht erreichen.

> **Note**
>
> Ein benutzerdefinierter Domainname muss innerhalb einer Region für alle AWS Konten eindeutig sein.
> Um einen Edge-optimierten benutzerdefinierten Domainnamen zwischen Regionen oder AWS Konten zu verschieben, müssen Sie die bestehende CloudFront Distribution löschen und eine neue erstellen. Der Vorgang kann etwa 30 Minuten dauern, bis der neue benutzerdefinierte Domainname verfügbar ist. Weitere Informationen finden Sie unter Aktualisieren von CloudFront -Verteilungen.

Edge-optimierte benutzerdefinierte Domänennamen

Wenn Sie eine Edge-optimierte API bereitstellen, richtet API Gateway eine CloudFront Amazon-Distribution und einen DNS-Eintrag ein, um den API-Domainnamen dem CloudFront Distributionsdomänennamen zuzuordnen. Anfragen für die API werden dann über die zugeordnete CloudFront Verteilung an API Gateway weitergeleitet.

Wenn Sie einen benutzerdefinierten Domainnamen für eine Edge-optimierte API erstellen, richtet API Gateway eine Verteilung ein CloudFront. Sie müssen jedoch einen DNS-Eintrag einrichten, um den benutzerdefinierten Domainnamen dem CloudFront Distributionsdomänennamen zuzuordnen. Diese Zuordnung ist für API-Anfragen vorgesehen, die daran gebunden sind, dass der benutzerdefinierte Domainname über die zugeordnete CloudFront Distribution an API Gateway weitergeleitet wird. Außerdem müssen Sie ein Zertifikat für den benutzerdefinierten Domänennamen bereitstellen.

> **Note**
>
> Die von API Gateway erstellte CloudFront Distribution gehört einem regionsspezifischen Konto, das mit API Gateway verbunden ist. Bei der Ablaufverfolgung von Vorgängen zur Erstellung und Aktualisierung einer solchen CloudFront Verteilung in CloudWatch Logs

müssen Sie diese API Gateway Gateway-Konto-ID verwenden. Weitere Informationen finden Sie unter Protokollieren Sie die Erstellung eines benutzerdefinierten Domainnamens CloudTrail.

Um einen Edge-optimierten benutzerdefinierten Domainnamen einzurichten oder das zugehörige Zertifikat zu aktualisieren, benötigen Sie die Berechtigung, Distributionen zu aktualisieren. CloudFront

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Die folgenden Berechtigungen sind erforderlich, um CloudFront Distributionen zu aktualisieren.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

API Gateway unterstützt Edge-optimierte benutzerdefinierte Domainnamen, indem es Server Name Indication (SNI) für die Verteilung nutzt. CloudFront Weitere Informationen zur Verwendung benutzerdefinierter Domainnamen in einer CloudFront Distribution, einschließlich des erforderlichen Zertifikatsformats und der maximalen Länge eines Zertifikatsschlüssels, finden Sie unter Verwenden alternativer Domainnamen und HTTPS im Amazon CloudFront Developer Guide.

Wenn Sie einen benutzerdefinierten Domänennamen als API-Host-Namen einrichten möchten, müssen Sie als Eigentümer der API ein SSL-/TLS-Zertifikat für den benutzerdefinierten Domänennamen bereitstellen.

Um ein Zertifikat für einen Edge-optimierten benutzerdefinierten Domänennamen bereitzustellen, können Sie AWS Certificate Manager (ACM) anweisen, ein neues Zertifikat in ACM zu generieren oder ein von der Zertifizierungsstelle eines Drittanbieters in der `us-east-1`-Region (USA Ost (Nord-Virginia)) ausgestelltes Zertifikat in ACM zu importieren.

Regionale benutzerdefinierte Domänennamen

Wenn Sie einen benutzerdefinierten Domänennamen für eine regionale API erstellen, erstellt API Gateway einen regionalen Domänennamen für die API. Sie müssen einen DNS-Datensatz so einrichten, dass der benutzerdefinierte Domänenname dem regionalen Domänennamen zugeordnet wird. Außerdem müssen Sie ein Zertifikat für den benutzerdefinierten Domänennamen bereitstellen.

Benutzerdefinierte Domänennamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domänennamen können Sie eine nahezu unendliche Anzahl von Domänennamen unterstützen, ohne das Standardkontingentzu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domänennamen geben, *`customername`*`.api.example.com`.

Um einen benutzerdefinierten Platzhalterdomänennamen zu erstellen, geben Sie einen Platzhalter (*) als erste Subdomäne einer benutzerdefinierten Domäne an, die alle möglichen Subdomänen einer Root-Domäne darstellt.

Der benutzerdefinierte Domänenname mit Platzhalter `*.example.com` führt beispielsweise zu Unterdomänen wie `a.example.com`, `b.example.com` und `c.example.com`, die alle zur gleichen Domäne weiterleiten.

Benutzerdefinierte Domänennamen mit Platzhaltern unterstützen andere Konfigurationen als die benutzerdefinierten Standarddomänennamen von API Gateway. In einem einzelnen AWS Konto können Sie beispielsweise unterschiedliche Konfigurationen vornehmen `*.example.com` und `a.example.com` sich unterschiedlich verhalten.

Sie können die Kontextvariablen `$context.domainName` und `$context.domainPrefix` verwenden, um den Domänennamen zu bestimmen, mit dem ein Client Ihre API aufruft. Weitere Informationen zu Kontextvariablen finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen.

Um einen benutzerdefinierten Domänennamen mit Platzhalter zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat angeben, das mithilfe der DNS- oder der E-Mail-Validierungsmethode validiert wurde.

> **Note**
>
> Sie können keinen benutzerdefinierten Platzhalter-Domänennamen erstellen, wenn ein anderes AWS Konto einen benutzerdefinierten Domänennamen erstellt hat, der mit dem benutzerdefinierten Platzhalter-Domänennamen in Konflikt steht. Wurde `a.example.com` beispielsweise von Konto A erstellt, dann kann Konto B als benutzerdefinierten Domänennamen mit Platzhalter nicht `*.example.com` erstellen.
> Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das AWS Supportcenter wenden, um eine Ausnahme anzufordern.

Zertifikate für benutzerdefinierte Domänennamen

> **Important**
>
> Sie geben das Zertifikat für Ihren benutzerdefinierten Domänennamen an. Wenn Ihre Anwendung Zertifikat-Pinning, manchmal auch SSL-Pinning genannt, verwendet, um ein ACM-Zertifikat anzuheften, kann die Anwendung nach der Verlängerung des Zertifikats möglicherweise keine Verbindung zu Ihrer Domain herstellen. AWS Weitere Informationen finden Sie unter Probleme beim Zertifikats-Pinning im AWS Certificate Manager - Benutzerhandbuch.

Um ein Zertifikat für einen benutzerdefinierten Domänennamen in einer Region bereitzustellen, in der ACM unterstützt wird, müssen Sie ein Zertifikat von ACM anfordern. Um ein Zertifikat für einen regionalen benutzerdefinierten Domänennamen in einer von ACM nicht unterstützten Region bereitzustellen, müssen Sie ein Zertifikat für API Gateway in dieser Region importieren.

Wenn Sie ein SSL-/TLS-Zertifikat importieren möchten, müssen Sie den PEM-formatierten Hauptteil des SSL-/TLS-Zertifikats, den persönlichen Schlüssel und die Zertifikatkette für den benutzerdefinierten Domänennamen bereitstellen. Ein in ACM gespeichertes Zertifikat wird durch seinen ARN angegeben. Um ein AWS verwaltetes Zertifikat für einen Domainnamen zu verwenden, verweisen Sie einfach auf dessen ARN.

Die Einrichtung und Verwendung eines benutzerdefinierten Domänennamens für eine API in ACM ist ganz einfach. Sie erstellen ein Zertifikat für den angegebenen Domänennamen (oder importieren ein Zertifikat), richten den Domänennamen in API Gateway mit dem ARN des von ACM bereitgestellten Zertifikats ein und ordnen einen Basispfad unter dem benutzerdefinierten Domänennamen einer Bereitstellungsphase der API zu. Wenn Sie Zertifikate verwenden, die von ACM ausgestellt werden, brauchen sich keine Gedanken über vertrauliche Zertifikatdetails wie private Schlüssel zu machen.

Themen

Vorbereiten von Zertifikaten in AWS Certificate Manager

Vor der Einrichtung eines benutzerdefinierten Domänennamens für eine API müssen Sie ein SSL-/TLS-Zertifikat in vorbereite AWS Certificate Manager. In den folgenden Schritten wird beschrieben, wie Sie dies tun. Weitere Informationen finden Sie im AWS Certificate Manager -Benutzerhandbuch.

> **Note**
>
> Um ein ACM-Zertifikat mit einem Edge-optimierten benutzerdefinierten API Gateway-Domänennamen zu verwenden, müssen Sie das Zertifikat in der Region `us-east-1` (USA Ost (Nord-Virginia)) anfordern oder importieren. Bei einem regionalen benutzerdefinierten API Gateway-Domänennamen müssen Sie das Zertifikat in derselben Region wie Ihre API anfordern oder importieren. Das Zertifikat muss von einer öffentlich vertrauenswürdigen Zertifizierungsstelle signiert sein und den benutzerdefinierten Domänennamen abdecken.

Registrieren Sie zunächst Ihre Internet-Domäne, zum Beispie, *`example`*`.com`. Sie können entweder Amazon Route 53 oder eine andere akkreditierte Domänenvergabestelle eines Drittanbieters verwenden. Eine Liste solcher Vergabestellen finden Sie unter Accredited Registrar Directory (Verzeichnis autorisierter Vergabestellen) auf der ICANN-Website.

Führen Sie die folgenden Schritte aus, um ein SSL/TLS-Zertifikat für einen Domänennamen in ACM zu erstellen oder zu importieren:

So fordern Sie ein von ACM bereitgestelltes Zertifikat für einen Domänennamen an

1. Melden Sie sich an der AWS Certificate Manager -Konsole an.
2. Wählen Sie Request a certificate aus.
3. Geben Sie einen benutzerdefinierten Domänennamen für Ihre API ein, z. B. `api.example.com` unter Domain name (Domänenname).
4. Wählen Sie optional Add another name to this certificate.
5. Wählen Sie Review and request.
6. Wählen Sie Confirm and request.
7. Damit eine Anforderung gültig ist, muss der registrierte Besitzer einer Internetdomäne der Anforderung vor der Ausstellung des Zertifikats durch ACM zustimmen.

So importieren Sie ein Zertifikat für einen Domänennamen in ACM

1. Fordern Sie bei einer Zertifizierungsstelle ein PEM-kodiertes SSL-/TLS-Zertifikat für den benutzerdefinierten Domänennamen an. Eine unvollständige Liste solcher Zertifizierungsstellen finden Sie unter Mozilla Included CA List

a. Erstellen Sie einen privaten Schlüssel für das Zertifikat und speichern Sie die Ausgabe in einer Datei; verwenden Sie dazu das OpenSSL-Toolkit auf der OpenSSL-Website:

```
openssl genrsa -out private-key-file 2048
```

> **Note**
>
> Amazon API Gateway nutzt Amazon CloudFront zur Unterstützung von Zertifikaten für benutzerdefinierte Domainnamen. Daher werden die Anforderungen und Einschränkungen eines benutzerdefinierten SSL/TLS-Zertifikats für Domainnamen von bestimmt. CloudFront Beispielsweise beträgt die maximale Länge des öffentlichen Schlüssels 2048 und die Länge des privaten Schlüssels 1024, 2048 und 4096. Die Länge des öffentlichen Schlüssels wird von der ausgewählten Zertifizierungsstelle bestimmt. Wenden Sie sich an Ihre Zertifizierungsstelle, um Schlüssel zurückzugeben, deren Länge vom Standard abweicht. Weitere Informationen finden Sie unter Sicherer Zugriff auf Ihre Objekte und Erstellen signierter URLs und signierter Cookies.

b. Erstellen Sie mithilfe von OpenSSL eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit dem zuvor erstellten privaten Schlüssel:

```
openssl req -new -sha256 -key private-key-file -out CSR-file
```

c. Senden Sie die Zertifikatssignierungsanforderung an die Zertifizierungsstelle und speichern Sie das Zertifikat.

d. Laden Sie die Zertifikatkette bei der Zertifizierungsstelle herunter.

> **Note**
>
> Wenn Ihnen der private Schlüssel auf andere Art bereitgestellt wird und er verschlüsselt ist, können Sie ihn mit dem folgenden Befehl entschlüsseln, bevor Sie ihn zur Einrichtung eines benutzerdefinierten Domänennamens an API Gateway senden.
>
> ```
> openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
> nocrypt -out MyDecryptedKey.pem
> ```

2. Laden Sie das Zertifikat hoch auf: AWS Certificate Manager

   a. Melden Sie sich an der AWS Certificate Manager -Konsole an.

   b. Wählen Sie Import a certificate.

   c. Geben oder fügen Sie für Certificate body (Zertifikatstext) den Text des PEM-formatierten Serverzertifikats ein, das Sie von Ihrer Zertifizierungsstelle erhalten haben. Im Folgenden sehen Sie ein verkürztes Beispiel eines solchen Zertifikats.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

   d. Geben oder fügen Sie für Certificate private key (Privater Schlüssel des Zertifikats) den privaten Schlüssel Ihres PEM-formatierten Zertifikats ein. Im Folgenden sehen Sie ein verkürztes Beispiel eines solchen Schlüssels.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO
...
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

   e. Geben oder fügen Sie unter Certificate chain (Zertifikatkette) nacheinander und ohne Leerzeilen die PEM-formatierten Zwischenzertifikate und optional das Stammzertifikat ein. Wenn Sie das Stammzertifikat eingeben, muss die Zertifikatkette mit Zwischenzertifikaten beginnen und mit dem Stammzertifikat enden. Verwenden Sie die von der Zertifizierungsstelle bereitgestellten Zwischenzertifikate. Verwenden Sie keine Zwischenzertifikate außerhalb der Vertrauenskette. Das folgende Beispiel zeigt ein verkürztes Beispiel.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifBlIK3se2e4/hEfcEejX/arxbx1BJCHBvlEPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

      Im Folgenden ein weiteres Beispiel.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Wählen Sie Review and import.

Wenn das Zertifikat erfolgreich erstellt oder importiert wurden, notieren Sie seinen ARN. Diesen benötigen Sie für die Einrichtung des benutzerdefinierten Domänennamens.

Auswahl einer Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway

Für mehr Sicherheit Ihrer benutzerdefinierten Amazon API Gateway Gateway-Domain können Sie eine Sicherheitsrichtlinie in der API Gateway-Konsole AWS CLI, dem oder einem AWS SDK auswählen.

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus TLS-Mindestversion und Verschlüsselungssammlungen, die von API Gateway angeboten werden. Sie können eine Sicherheitsrichtlinie mit TLS-Version 1.2 oder TLS-Version 1.0 wählen. Das TLS-Protokoll behandelt Netzwerksicherheitsprobleme wie Manipulationen und Abhören zwischen einem Client und einem Server. Wenn Ihre Clients über die benutzerdefinierte Domäne einen TLS-Handshake mit Ihrer API ausführen, erzwingt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite-Optionen, die von Ihren Clients verwendet werden können.

In benutzerdefinierten Domäneneinstellungen legt eine Sicherheitsrichtlinie zwei Einstellungen fest:

- Die TLS-Mindestversion, die API Gateway für die Kommunikation mit API-Clients verwendet.
- Das Verschlüsselungsverfahren, das API Gateway für die Verschlüsselung des Inhalts verwendet, den es an die API-Clients zurückgibt.

Wenn Sie eine TLS 1.0-Sicherheitsrichtlinie wählen, akzeptiert die Sicherheitsrichtlinie TLS 1.0-, TLS 1.2- und TLS 1.3-Verkehr. Wenn Sie eine TLS 1.2-Sicherheitsrichtlinie wählen, akzeptiert die Sicherheitsrichtlinie TLS 1.2- und TLS 1.3-Verkehr und lehnt TLS 1.0-Verkehr ab.

> **Note**
>
> Sie können eine Sicherheitsrichtlinie nur für eine benutzerdefinierte Domain angeben. Für eine API, die einen Standardendpunkt verwendet, verwendet API Gateway die folgende Sicherheitsrichtlinie:
>
> - Für Edge-optimierte APIs: `TLS-1-0`
> - Für regionale APIs: `TLS-1-0`
> - Für private APIs: `TLS-1-2`

Themen

Wie spezifiziert man eine Sicherheitsrichtlinie für benutzerdefinierte Domains

Wenn Sie einen benutzerdefinierten Domänennamen erstellen, geben Sie die Sicherheitsrichtlinie für diesen an. Informationen zum Erstellen einer benutzerdefinierten Domäne finden Sie unter the section called “Erstellen eines Edge-optimierten benutzerdefinierten Domänennamens” oderthe section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

Um die Sicherheitsrichtlinie für Ihren benutzerdefinierten Domainnamen zu ändern, aktualisieren Sie die benutzerdefinierten Domäneneinstellungen. Sie können Ihre benutzerdefinierten Domainnamen-Einstellungen mit dem AWS Management Console AWS CLI, dem oder einem AWS SDK aktualisieren.

Wenn Sie die API Gateway REST-API oder verwenden AWS CLI, geben Sie die neue TLS-Version `TLS_1_0` oder `TLS_1_2` im `securityPolicy` Parameter an. Weitere Informationen finden Sie unter

domainname:update in der Amazon API Gateway REST API-Referenz oder update-domain-namein der AWS CLI Referenz.

Der Aktualisierungsvorgang kann einige Minuten dauern.

Unterstützte Sicherheitsrichtlinien, TLS-Protokollversionen und Verschlüsselungen für Edge-optimierte benutzerdefinierte Domänen

In der folgenden Tabelle werden die Sicherheitsrichtlinien beschrieben, die für Edge-optimierte benutzerdefinierte Domänennamen angegeben werden können.

Sicherheitsrichtlinie	TLS_1_0	TLS_1_2
TLS-Protokolle		
TLSv1.3	♦	♦
TLSv1.2	♦	♦
TLSv1.1	♦	
TLSv1	♦	
TLS-Chiffren		
TLS_AES_128_GCM_SHA256	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA	♦	
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦

Sicherheitsrichtlinie	TLS_1_0	TLS_1_2
ECDHE-ECDSA-CHACHA20-POLY1305	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦
ECDHE-ECDSA-AES256-SHA	♦	
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA	♦	
ECDHE-RSA-AES256-GCM-SHA384	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA	♦	
AES128-GCM-SHA256	♦	
AES256-GCM-SHA384	♦	♦
AES128-SHA256	♦	♦
AES256-SHA	♦	
AES128-SHA	♦	
DES-CBC3-SHA	♦	

Unterstützte Sicherheitsrichtlinien, TLS-Protokollversionen und Chiffren für regionale benutzerdefinierte Domänen

In der folgenden Tabelle werden die Sicherheitsrichtlinien beschrieben, die für regionale benutzerdefinierte Domainnamen angegeben werden können.

Sicherheitsrichtlinie	TLS_1_0	TLS_1_2
TLS-Protokolle		
TLSv1.3	♦	♦
TLSv1.2	♦	♦
TLSv1.1	♦	
TLSv1	♦	
TLS-Chiffren		
TLS_AES_128_GCM_SHA256	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA	♦	

Sicherheitsrichtlinie	TLS_1_0	TLS_1_2
ECDHE-RSA-AES128-SHA	♦	
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦
ECDHE-RSA-AES256-GCM-SHA384	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA	♦	
ECDHE-ECDSA-AES256-SHA	♦	
AES128-GCM-SHA256	♦	♦
AES128-SHA256	♦	♦
AES128-SHA	♦	
AES256-GCM-SHA384	♦	♦
AES256-SHA256	♦	♦
AES256-SHA	♦	

Unterstützte TLS-Protokollversionen und Chiffren für private APIs

In der folgenden Tabelle werden das unterstützte TLS-Protokoll und die Verschlüsselungen für private APIs beschrieben. Die Angabe einer Sicherheitsrichtlinie für private APIs wird nicht unterstützt.

Sicherheitsrichtlinie	TLS_1_2
TLS-Protokolle	
TLSv1.2	♦
TLS-Chiffren	
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für dieselben Chiffren. Die folgende Tabelle ordnet den OpenSSL-Namen dem RFC-Namen für jedes Verschlüsselungsverfahren zu.

OpenSSL-Verschlüsselungsname	RFC-Verschlüsselungsname
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256

OpenSSL-Verschlüsselungsname	RFC-Verschlüsselungsname
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Informationen zu HTTP-APIs und APIs WebSocket

Weitere Informationen zu HTTP-APIs und WebSocket APIs finden Sie unter the section called “Sicherheitsrichtlinie für HTTP-APIs” undthe section called “Sicherheitsrichtlinie für WebSocket APIs”.

Erstellen eines Edge-optimierten benutzerdefinierten Domänennamens

Themen

- Einrichten eines Edge-optimierten benutzerdefinierten Domänennamens für eine API Gateway-API
- Protokollieren Sie die Erstellung eines benutzerdefinierten Domainnamens CloudTrail
- Konfigurieren eines Basispfad-Mappings einer API mit einem benutzerdefinierten Domänennamen als Hostname
- Wechseln eines in ACM importierten Zertifikats
- Aufrufen Ihrer API mit benutzerdefinierten Domänennamen

Einrichten eines Edge-optimierten benutzerdefinierten Domänennamens für eine API Gateway-API

Im folgenden Verfahren wird beschrieben, wie Sie einen benutzerdefinierten Domain-Namen für eine API mit der API Gateway-Konsole erstellen.

So erstellen Sie einen benutzerdefinierten Domänennamen mit der API Gateway-Konsole

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Benutzerdefinierten Domänennamen im Hauptnavigationsbereich aus.
3. Wählen Sie Create aus.
4. Geben Sie für Domänenname einen Domänennamen ein.
5. Wählen Sie unter Konfiguration die Option Edge-optimiert aus.
6. Wählen Sie eine minimale TLS-Version aus.
7. Wählen Sie ein ACM-Zertifikat aus.

   > **Note**
   >
   > Um ein ACM-Zertifikat mit einem Edge-optimierten benutzerdefinierten API Gateway-Domänennamen zu verwenden, müssen Sie das Zertifikat in der Region `us-east-1` (USA Ost (Nord-Virginia)) anfordern oder importieren.

8. Wählen Sie Domain-Namen erstellen aus.
9. Nachdem der benutzerdefinierte Domainname erstellt wurde, zeigt die Konsole den zugehörigen CloudFront Distributionsdomänennamen in der Form von *`distribution-id`*`.cloudfront.net` zusammen mit dem Zertifikat-ARN an. Notieren Sie sich den Namen der CloudFront Vertriebsdomäne, der in der Ausgabe angezeigt wird. Sie benötigen ihn im nächsten Schritt zur Festlegung des CNAME-Werts oder des Aliasziels für den A-Datensatz des benutzerdefinierten Domänennamens im DNS.

   > **Note**
   >
   > Die Erstellung des neuen benutzerdefinierten Domänennamen dauert ca. 40 Minuten. In der Zwischenzeit können Sie den DNS-Eintragsalias so konfigurieren, dass der benutzerdefinierte Domainname dem zugehörigen CloudFront Distributionsdomänennamen zugeordnet wird und dass die Basispfadzuordnung für den benutzerdefinierten Domainnamen eingerichtet wird, während der benutzerdefinierte Domainname initialisiert wird.

10. Als Nächstes konfigurieren Sie DNS-Einträge bei Ihrem DNS-Anbieter, um den benutzerdefinierten Domainnamen der zugehörigen CloudFront Distribution zuzuordnen.

Anweisungen für Amazon Route 53 finden Sie unter Weiterleiten des Datenverkehrs an eine Amazon API Gateway API unter Verwendung Ihres Domain-Namens im Amazon Route 53-Entwicklerhandbuch.

Bei den meisten DNS-Anbietern wird der gehosteten Zone ein benutzerdefinierter Domänenname als CNAME-Ressourcendatensatz hinzugefügt. Der Name des CNAME-Datensatzes entspricht dem von Ihnen zuvor unter Domänenname eingegebenen benutzerdefinierten Domänennamen (z. B. `api.example.com`). Der CNAME-Eintragswert gibt den Domainnamen für die CloudFront Verteilung an. Mit einem CNAME-Datensatz funktioniert dies allerdings nicht, wenn es sich bei Ihrer benutzerdefinierten Domäne um einen Zone Apex handelt (d. h. `example.com` und nicht `api.example.com`). Ein Zone Apex wird auch als Stammdomäne Ihrer Organisation bezeichnet. Für einen Zone Apex müssen Sie einen A-Datensatzalias verwenden, vorausgesetzt, dieser wird von Ihrem DNS-Anbieter unterstützt.

Mit Route 53 können Sie einen A-Record-Alias für Ihren benutzerdefinierten Domainnamen erstellen und den CloudFront Distributionsdomänennamen als Alias-Ziel angeben. Das bedeutet, dass Route 53 den benutzerdefinierten Domänennamen auch dann weiterleiten kann, wenn es sich um einen Zone Apex handelt. Weitere Informationen finden Sie im Amazon Route 53-Entwicklerhandbuch unter Wählen zwischen Alias- und Nicht-Alias-Ressourcendatensätzen.

Durch die Verwendung von A-Record-Aliasnamen wird außerdem verhindert, dass der zugrunde liegende CloudFront Vertriebsdomänenname offengelegt wird, da die Zuordnung von Domainnamen ausschließlich innerhalb von Route 53 erfolgt. Aus diesen Gründen empfehlen wir die Verwendung von Route 53-A-Datensatzaliasnamen.

Zusätzlich zur API Gateway-Konsole können Sie die API Gateway REST API, AWS CLI oder eines der AWS SDKs verwenden, um den benutzerdefinierten Domainnamen für Ihre APIs einzurichten. Zur Veranschaulichung wird das Verfahren im Folgenden unter Verwendung der REST-API-Aufrufe beschrieben.

So richten Sie einen benutzerdefinierten Domänennamen mit der API Gateway-REST-API ein

1. Rufen Sie die Methode domainname:create auf und geben Sie den benutzerdefinierten Domänennamen und den ARN eines in AWS Certificate Manager gespeicherten Zertifikats an.

   Der erfolgreiche API-Aufruf gibt eine `201 Created` Antwort zurück, die den Zertifikat-ARN sowie den zugehörigen CloudFront Distributionsnamen in der Nutzlast enthält.

2. Notieren Sie sich den Namen der CloudFront Distributionsdomäne, der in der Ausgabe angezeigt wird. Sie benötigen ihn im nächsten Schritt zur Festlegung des CNAME-Werts oder des Aliasziels für den A-Datensatz des benutzerdefinierten Domänennamens im DNS.
3. Gehen Sie wie zuvor beschrieben vor, um einen A-Record-Alias einzurichten, um den benutzerdefinierten Domainnamen seinem CloudFront Distributionsnamen zuzuordnen.

Code-Beispiele für diesen REST-API-Aufruf finden Sie unter domainname:create.

Protokollieren Sie die Erstellung eines benutzerdefinierten Domainnamens CloudTrail

Wenn für die Protokollierung von API-Gateway-Aufrufen, die von Ihrem Konto getätigt wurden, aktiviert CloudTrail ist, protokolliert API Gateway die zugehörigen CloudFront Distributionsupdates, wenn ein benutzerdefinierter Domainname für eine API erstellt oder aktualisiert wird. Da diese CloudFront Distributionen Eigentum von API Gateway sind, wird jede dieser gemeldeten CloudFront Distributionen durch eine der folgenden regionsspezifischen API Gateway Gateway-Konto-IDs und nicht durch die Konto-ID des API-Besitzers identifiziert.

Region	Konto-ID
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161

Region	Konto-ID
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Konfigurieren eines Basispfad-Mappings einer API mit einem benutzerdefinierten Domänennamen als Hostname

Sie können einen einzelnen benutzerdefinierten Domänennamen als Hostnamen mehrerer APIs verwenden. Dies erreichen Sie durch Konfigurieren der Basispfad-Mappings im benutzerdefinierten Domänennamen. Mit den Basispfad-Mappings kann auf eine API, die sich unter der benutzerdefinierten Domäne befindet, über die Kombination aus benutzerdefiniertem Domänennamen und zugehörigem Basispfad zugegriffen werden.

Wenn Sie beispielsweise eine API mit dem Namen `PetStore` und eine weitere mit dem Namen `PetShop` erstellt haben und den benutzerdefinierten Domänennamen `api.example.com` in API Gateway einrichten, können Sie als URL der API `PetStore` wahlweise `https://api.example.com` oder `https://api.example.com/myPetStore` festlegen. Die API `PetStore` ist mit dem Basispfad einer leeren Zeichenfolge oder `myPetStore` unterhalb des benutzerdefinierten Domänennamens `api.example.com` verknüpft. Gleichermaßen können Sie einen Basispfad `yourPetShop` für die API `PetShop` zuweisen. Die URL `https://api.example.com/yourPetShop` ist dann die Stamm-URL der API `PetShop`.

Bevor Sie den Basispfad für eine API festlegen, führen Sie die Schritte unter au Einrichten eines Edge-optimierten benutzerdefinierten Domänennamens für eine API Gateway-API.

Das folgende Verfahren richtet API-Mappings ein, um Pfade aus Ihrem benutzerdefinierten Domänennamen Ihren API-Stufen zuzuordnen.

So erstellen Sie API-Zuordnungen mit der API Gateway Gateway-Konsole

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie einen benutzerdefinierten Domänennamen aus.
3. Wählen Sie API-Mappings konfigurieren aus.
4. Wählen Sie Neues Mapping hinzufügen aus.
5. Geben Sie die API, die Stufe und den Pfad (optional) für das Mapping an.
6. Wählen Sie Save (Speichern) aus.

Darüber hinaus können Sie die API Gateway REST API, AWS CLI oder eines der AWS SDKs aufrufen, um die Basispfadzuordnung einer API mit einem benutzerdefinierten Domainnamen als Hostnamen einzurichten. Zur Veranschaulichung wird das Verfahren im Folgenden unter Verwendung der REST-API-Aufrufe beschrieben.

So richten Sie das Basispfad-Mapping einer API mit der API Gateway-REST-API ein

- Rufen Sie die Methode basepathmapping:create für einen bestimmten benutzerdefinierten Domänennamen auf und geben Sie in der Nutzlast der Anforderung die `basePath`- und `restApiId`-Eigenschaft sowie eine `stage`-Eigenschaft für die Bereitstellung an.

  Ein erfolgreicher API-Aufruf gibt die Antwort `201 Created` zurück.

Codebeispiele des REST-API-Aufrufs finden Sie unter basepathmapping:create.

Wechseln eines in ACM importierten Zertifikats

ACM übernimmt automatisch die Verlängerung eigener ausgestellter Zertifikate. Sie müssen keine von ACM ausgestellten Zertifikate für Ihre benutzerdefinierten Domainnamen rotieren. CloudFront kümmert sich in Ihrem Namen darum.

Wenn Sie jedoch ein Zertifikat in ACM importieren und für einen benutzerdefinierten Domänennamen verwenden, müssen Sie das Zertifikat vor seinem Ablauf auswechseln. Dies umfasst das Importieren eines neuen Drittanbieter-Zertifikats für den Domänennamen und das Auswechseln des vorhandenen durch ein neues Zertifikat. Diesen Vorgang müssen Sie wiederholen, wenn das neu importierten Zertifikat abläuft. Alternativ können Sie in ACM die Ausstellung eines neuen Zertifikats für den Domänennamen beauftragen und das vorhandene durch das neue auswechseln. Danach können Sie ACM verlassen und CloudFront die Zertifikatsrotation automatisch für Sie übernehmen. Wenn Sie ein neues ACM-Zertifikat erstellen oder importieren möchten, führen Sie die Schritte zum Anfordern oder Importieren neuer ACM-Zertifikate für den jeweiligen Domänennamen aus.

Um ein Zertifikat für einen Domainnamen zu rotieren, können Sie die API Gateway Gateway-Konsole, die API Gateway Gateway-REST-API, AWS CLI oder eines der AWS SDKs verwenden.

So tauschen Sie ein ablaufendes Zertifikat, das in ACM importiert wurde, mit der API Gateway-Konsole aus

1. Fordern Sie ein Zertifikat in ACM an oder importieren Sie es.
2. Gehen Sie zurück zur API Gateway-Konsole.
3. Wählen Sie Custom domain names (Benutzerdefinierte Domänennamen) im Hauptnavigationsbereich der API Gateway-Konsole aus.
4. Wählen Sie einen benutzerdefinierten Domänennamen aus.
5. Wählen Sie Edit aus.
6. Wählen Sie das gewünschte Zertifikat aus der Dropdown-Liste ACM-Zertifikat aus.
7. Klicken Sie auf Speichern, um mit dem Austausch des Zertifikats für den benutzerdefinierten Domänennamen zu beginnen.

> **Note**
>
> Es dauert ca. 40 Minuten, bis der Prozess abgeschlossen ist. Nach Abschluss des Austauschvorgangs können Sie durch Klicken auf den Doppelpfeil neben dem ACM Certificate das ursprüngliche Zertifikat wiederherstellen.

Im Folgenden wird der programmgesteuerte Austausch eines importieren Zertifikats für einen benutzerdefinierten Domänennamen unter Verwendung der API Gateway-REST-API veranschaulicht.

Austauschen eines importierten Zertifikats mit der API Gateway-REST-API

- Rufen Sie die Aktion domainname:update unter Angabe des ARN des neuen ACM-Zertifikats für den jeweiligen Domänennamen auf.

Aufrufen Ihrer API mit benutzerdefinierten Domänennamen

Sofern die korrekte URL verwendet wird, macht es keinen Unterschied, ob eine API mit einem benutzerdefinierten oder dem standardmäßigen Domänennamen aufgerufen wird.

In den folgenden Beispielen werden mehrere Standard-URLs mit den entsprechenden benutzerdefinierten URLs zweier APIs (`udxjef` und `qf3duz`) in einer bestimmten Region (`us-east-1`) mit einem bestimmten benutzerdefinierten Domänennamen (`api.example.com`) verglichen.

Stamm-URLs von APIs mit standardmäßigen und benutzerdefinierten Domänennamen

API-ID	Stufe	Standard-URL	Basispfad	Custom URL
udxjef	Prod	https://udxjef.execute-api.us-east-1.amazonaws.com/prod	/petstore	https://api.example.com/petstore
udxjef	tst	https://udxjef.execute-api.us-east-1	/petdepot	https://api.example.com/petdepot

API-ID	Stufe	Standard-URL	Basispfad	Custom URL
		.amazonaws.com/tst		
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/bookstore	https://api.example.com/bookstore
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/bookstand	https://api.example.com/bookstand

API Gateway unterstützt benutzerdefinierte Domänennamen für eine API über die Server Name Indication (SNI). Sie können die API mit einem benutzerdefinierten Domänennamen über einen Browser oder eine Client-Bibliothek aufrufen, die SNI unterstützt.

API Gateway erzwingt SNI bei der CloudFront Distribution. Informationen zur CloudFront Verwendung benutzerdefinierter Domainnamen finden Sie unter Amazon CloudFront Custom SSL.

Einrichten eines regionalen benutzerdefinierten Domänennamens in API Gateway

Sie können einen benutzerdefinierten Domainnamen für einen regionalen API-Endpunkt (für eine AWS Region) erstellen. Um einen benutzerdefinierten Domänennamen zu erstellen, müssen Sie ein regionsspezifisches ACM-Zertifikat bereitstellen. Weitere Informationen zum Erstellen oder Hochladen eines benutzerdefinierten Domänennamen-Zertifikats finden Sie unter Vorbereiten von Zertifikaten in AWS Certificate Manager.

> **Important**
>
> Bei einem regionalen benutzerdefinierten API Gateway-Domänennamen müssen Sie das Zertifikat in derselben Region wie Ihre API anfordern oder importieren.

Wenn Sie einen regionalen benutzerdefinierten Domänennamen mit einem ACM-Zertifikat erstellen (oder migrieren), erstellt API Gateway eine serviceverknüpfte Rolle in Ihrem Konto, wenn die Rolle nicht bereits vorhanden ist. Die serviceverknüpfte Rolle ist erforderlich, um Ihr ACM-Zertifikat Ihrem regionalen Endpunkt anzuhängen. Die Rolle hat einen Namen AWSServiceRoleForAPIGatewayund ihr wird die GatewayServiceRolePolicyAPI-verwaltete Richtlinie zugewiesen. Weitere Informationen zur Verwendung von serviceverknüpften Rollen finden Sie unter Verwenden serviceverknüpfter Rollen.

> **Important**
>
> Sie müssen einen DNS-Datensatz so erstellen, dass der benutzerdefinierte Domänenname auf den regionalen Domänennamen verweist. Auf diese Weise kann der Datenverkehr, der an den benutzerdefinierten Domänennamen gebunden ist, an den regionalen Hostnamen der API weitergeleitet werden. Der DNS-Datensatz kann der Typ CNAME oder "A" sein.

Themen

Einrichten eines regionalen benutzerdefinierten Domänennamens mit einem ACM-Zertifikat über die API Gateway-Konsole

Im folgenden Verfahren wird beschrieben, wie Sie einen regionalen benutzerdefinierten Domänennamen mit der API Gateway-Konsole einrichten.

So richten Sie einen regionalen benutzerdefinierten Domänennamen mit der API Gateway-Konsole ein

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Benutzerdefinierten Domänennamen im Hauptnavigationsbereich aus.
3. Wählen Sie Create aus.
4. Geben Sie für Domänenname einen Domänennamen ein.

5. Wählen Sie unter Konfiguration die Option Regional aus.
6. Wählen Sie eine minimale TLS-Version aus.
7. Wählen Sie ein ACM-Zertifikat aus. Das Zertifikat muss sich in derselben Region wie die API befinden.
8. Wählen Sie Create aus.
9. Befolgen Sie die Route 53-Dokumentation für die Konfiguration von Route 53 zur Weiterleitung von Datenverkehr an API Gateway.

Das folgende Verfahren richtet API-Mappings ein, um Pfade aus Ihrem benutzerdefinierten Domänennamen Ihren API-Stufen zuzuordnen.

So erstellen Sie API-Zuordnungen mit der API Gateway Gateway-Konsole

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie einen benutzerdefinierten Domänennamen aus.
3. Wählen Sie API-Mappings konfigurieren aus.
4. Wählen Sie Neues Mapping hinzufügen aus.
5. Geben Sie die API, die Stufe und den Pfad für das Mapping an.
6. Wählen Sie Save (Speichern) aus.

Weitere Informationen zum Festlegen von Basispfad-Mappings für die benutzerdefinierte Domäne finden Sie unter Konfigurieren eines Basispfad-Mappings einer API mit einem benutzerdefinierten Domänennamen als Hostname.

Richten Sie einen regionalen benutzerdefinierten Domainnamen mit einem ACM-Zertifikat ein AWS CLI

Gehen Sie wie folgt vor AWS CLI , um mit dem einen benutzerdefinierten Domainnamen für eine regionale API einzurichten.

1. Rufen Sie `create-domain-name` auf, was einen benutzerdefinierten Domänennamen und den ARN eines regionalen Zertifikats angibt.

```
aws apigatewayv2 create-domain-name \
    --domain-name 'regional.example.com' \
```

```
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678
```

Beachten Sie, dass das angegebene Zertifikat aus der Region `us-west-2` stammt und für dieses Beispiel nehmen wir an, dass die zugrunde liegende API von der gleichen Region stammt.

Ist der Aufruf erfolgreich, wird ein Ergebnis zurückgegeben, das wie folgt aussehen sollte:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-id.execute-api.us-west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/id",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "id",
            "SecurityPolicy": "TLS_1_2"
        }
    ]
}
```

Der `DomainNameConfigurations`-Eigenschaftswert gibt den Hostnamen der regionalen API zurück. Sie müssen einen DNS-Datensatz so erstellen, dass Ihr benutzerdefinierter Domänenname auf diesen regionalen Domänennamen verweist. Auf diese Weise kann der Datenverkehr, der an den benutzerdefinierten Domänennamen gebunden ist, an diesen Hostnamen der regionalen API weitergeleitet werden.

2. Erstellen Sie einen DNS-Datensatz, um den benutzerdefinierten Domänennamen und den regionalen Domänennamen zu verknüpfen. Auf diese Weise können Anforderungen, die an den benutzerdefinierten Domänennamen gebunden sind, an den regionalen Hostnamen der API weitergeleitet werden.
3. Fügen Sie ein Basispfad-Mapping hinzu, um die angegebene API (z. B. `0qzs2sy7bh`) in einer Bereitstellungsphase (z. B. `test`) unter dem angegebenen benutzerdefinierten Domänennamen (z. B. `regional.example.com`) bereitzustellen.

```
aws apigatewayv2 create-api-mapping \
    --domain-name 'regional.example.com' \
```

```
    --api-mapping-key 'myApi' \
    --api-id 0qzs2sy7bh \
    --stage 'test'
```

Dies hat zur Folge, dass das Basis-URL mithilfe des benutzerdefinierten Domänennamens für die API, die in der Stufe bereitgestellt wird, zu wir `https://regional.example.com/myAPI`.

4. Konfigurieren Sie Ihre DNS-Datensätze so, dass der regionale benutzerdefinierte Domänenname seinem Hostnamen in der ID der gehosteten Zone zugewiesen wird. Erstellen Sie zunächst eine JSON-Datei, die die Konfiguration für die Einrichtung eines DNS-Datensatzes für den regionalen Domänennamen enthält. Das folgende Beispiel zeigt, wie Sie einen DNS-A-Datensatz erstellen, um einen regionalen benutzerdefinierten Domänennamen (`regional.example.com`) seinem regionalen Hostnamen zuzuweisen (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`), der als Teil der benutzerdefinierten Domänenname-Erstellung bereitgestellt wurde. Die Eigenschaften `DNSName` und `HostedZoneId` von `AliasTarget` können die Werte `regionalDomainName` und `regionalHostedZoneId` des benutzerdefinierten Domänennamens entsprechend annehmen. Sie können die regionalen IDs der gehosteten Route 53-Zone auch in Amazon API Gateway-Endpunkte und -Kontingente abrufen.

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "regional.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

5. Führen Sie den folgenden CLI-Befehl aus:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id {your-hosted-zone-id} \
```

```
    --change-batch file://path/to/your/setup-dns-record.json
```

wobei *{your-hosted-zone-id}* die Route 53 Hosted Zone-ID des DNS-Eintrags ist, der in Ihrem Konto festgelegt ist. Der `change-batch` Parameterwert verweist auf eine JSON-Datei (*setup-dns-record.json*) in einem Ordner (*path/to/your*).

Migrieren eines benutzerdefinierten Domänennamens in einen anderen API-Endpunkt

Sie können Ihren benutzerdefinierten Domänennamen zwischen Edge-optimierten und regionalen Endpunkten migrieren. Zunächst fügen Sie den neuen Endpunktkonfigurationstyp zur bestehenden Liste `endpointConfiguration.types` für den benutzerdefinierten Domänennamen hinzu. Danach richten Sie einen DNS-Datensatz so ein, dass der benutzerdefinierte Domänenname auf den neu bereitgestellten Endpunkt verweist. Optimal können Sie dann noch die Konfigurationsdaten des veralteten benutzerdefinierten Domänennamens löschen.

Beachten Sie bei der Migration von einem Edge-optimierten benutzerdefinierten Domänennamen, dass das von ACM bereitgestellte erforderliche Zertifikat aus der Region (USA Ost (Nord-Virginia)) stammen m (`us-east-1`). Dieses Zertifikat wird auf alle geografischen Standorte verteilt. Bei einer regionalen API muss sich das ACM-Zertifikat für den regionalen Domänennamen dagegen in derselben Region wie die API befinden. Sie können einen Edge-optimierten benutzerdefinierten Domänennamen, der sich außerhalb der Region `us-east-1` befindet, zu einem regionalen benutzerdefinierten Domänennamen migrieren, indem Sie zunächst ein neues ACM-Zertifikat für die lokale Region der API anfordern.

Die Migration zwischen einem Edge-optimierten und einem regionalen benutzerdefinierten Domänennamen in API Gateway kann bis zu 60 Sekunden dauern. Wann der neu erstellte Endpunkt Traffic akzeptieren kann, hängt die Migrationszeit auch davon ab, wann Sie Ihre DNS-Datensätze aktualisieren.

Themen

- Migrieren Sie benutzerdefinierte Domainnamen mit dem AWS CLI

Migrieren Sie benutzerdefinierte Domainnamen mit dem AWS CLI

Um einen benutzerdefinierten Domainnamen von einem Edge-optimierten Endpunkt zu einem regionalen Endpunkt oder umgekehrt zu migrieren, rufen Sie den `update-domain-name`Befehl auf, um den neuen Endpunkttyp hinzuzufügen, und rufen Sie optional den `update-domain-name`Befehl auf, um den alten Endpunkttyp zu entfernen. AWS CLI

Themen

Migrieren eines Edge-optimierten benutzerdefinierten Domänennamens in einen regionalen benutzerdefinierten Domänennamen

Rufen Sie den CLI-Befehl `update-domain-name` wie folgt auf, um einen Edge-optimierten benutzerdefinierten Domänennamen in einen regionalen benutzerdefinierten Domänennamen zu migrieren:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  [ \
        { op:'add', path: '/endpointConfiguration/types',value: 'REGIONAL' }, \
        { op:'add', path: '/regionalCertificateArn', value: 'arn:aws:acm:us-
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149' } \
      ]
```

Das regionale Zertifikat muss sich in derselben Region befinden wie die regionale API.

Die erfolgreiche Antwort verfügt über den Statuscode `200 OK` und einen Inhalt, der etwa wie folgt aussieht:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
cd833b28-58d2-407e-83e9-dce3fd852149",
```

```
    "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"
}
```

Für den migrierten regionalen benutzerdefinierten Domänennamen gibt die resultierende Eigenschaft `regionalDomainName` den regionalen API-Hostnamen zurück. Sie müssen einen DNS-Datensatz so einrichten, dass der regionale benutzerdefinierte Domänenname auf diesen regionalen Hostnamen verweist. Auf diese Weise kann der Datenverkehr, der an den benutzerdefinierten Domänennamen gebunden ist, an den regionalen Host weitergeleitet werden.

Nachdem der DNS-Eintrag eingerichtet wurde, können Sie den Edge-optimierten benutzerdefinierten Domainnamen entfernen, indem Sie den folgenden Befehl aufrufen: `update-domain-name` AWS CLI

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations [ \
            {op:'remove', path:'/endpointConfiguration/types', value:'EDGE'}, \
            {op:'remove', path:'certificateName'}, \
            {op:'remove', path:'certificateArn'} \
        ]
```

Migrieren eines regionalen benutzerdefinierten Domänennamens in einen Edge-optimierten benutzerdefinierten Domänennamen

Um einen benutzerdefinierten Regionaldomänennamen zu einem Edge-optimierten benutzerdefinierten Domainnamen zu migrieren, rufen Sie den `update-domain-name` Befehl von wie folgt auf AWS CLI:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  [ \
            { op:'add', path:'/endpointConfiguration/types',value: 'EDGE' }, \
            { op:'add', path:'/certificateName', value:'edge-cert'}, \
            { op:'add', path:'/certificateArn', value: 'arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a' } \
        ]
```

Das Zertifikat der Edge-optimierten Domäne muss in der Region `us-east-1` erstellt werden.

Die erfolgreiche Antwort verfügt über den Statuscode `200 OK` und einen Inhalt, der etwa wie folgt aussieht:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/3d881b54-851a-478a-a887-f6502760461d",
    "regionalDomainName": "d-cgkq2qwgzf.execute-api.us-east-1.amazonaws.com"
}
```

Für den angegebenen benutzerdefinierten Domänennamen gibt API Gateway den Edge-optimierten API-Hostnamen als Wert der Eigenschaft `distributionDomainName` zurück. Sie müssen einen DNS-Datensatz so einrichten, dass der Edge-optimierte benutzerdefinierte Domänenname auf diesen Verteilungsdomänennamen verweist. Auf diese Weise kann der Datenverkehr, der an den Edge-optimierten benutzerdefinierten Domänennamen gebunden ist, an den Edge-optimierten Hostnamen der API weitergeleitet werden.

Nachdem Sie den DNS-Datensatz eingerichtet haben, können Sie den Endpunkttyp `REGION` des benutzerdefinierten Domänennamens löschen:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations [ \
        {op:'remove', path:'/endpointConfiguration/types', value:'REGIONAL'}, \
        {op:'remove', path:'regionalCertificateArn'} \
      ]
```

Das Ergebnis dieses Befehls sieht in etwa wie folgende Ausgabe aus und enthält nur Konfigurationsdaten des Edge-optimierten Domänennamens:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
```

```
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "regional.haymuto.com",
    "endpointConfiguration": {
        "types": "EDGE"
    }
}
```

Arbeiten mit API-Mappings für REST-APIs

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Nachdem Sie einen Domain-Namen erstellt und DNS-Einträge konfiguriert haben, verwenden Sie API-Mappings, um Datenverkehr über Ihren benutzerdefinierten Domain-Namen an Ihre APIs zu senden.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise die `production`-Phase einer API in `https://api.example.com/orders` abbilden.

Sie können HTTP-API- und REST-API--Stufen demselben benutzerdefinierten Domain-Namen zuweisen.

Bevor Sie ein API-Mapping erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

Weiterleiten von API-Anforderungen

Sie können API-Mappings mit mehreren Ebenen konfigurieren, z. B. `orders/v1/items` und `orders/v2/items`.

> **Note**
>
> Um API-Mappings mit mehreren Ebenen zu konfigurieren, muss Ihr benutzerdefinierter Domain-Name regional sein und die Sicherheitsrichtlinie TLS 1.2 verwenden.

Bei API-Mappings mit mehreren Ebenen leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspfad hat. API Gateway berücksichtigt nur die für API-Mappings

konfigurierten Pfade und keine API-Routen, um die aufzurufende API auszuwählen. Wenn kein Pfad mit der Anforderung übereinstimmt, sendet API Gateway die Anforderung an die API, die Sie dem leeren Pfad zugeordnet habe (`none`).

Bei benutzerdefinierten Domain-Namen, die API-Mappings mit mehreren Ebenen verwenden, leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspräfix hat.

Betrachten Sie beispielsweise einen benutzerdefinierten Domain-Namen `https://api.example.com` mit den folgenden API-Mappings:

1. `(none)` API 1 zugewiesen.
2. `orders` API 2 zugewiesen.
3. `orders/v1/items` API 3 zugewiesen.
4. `orders/v2/items` API 4 zugewiesen.
5. `orders/v2/items/categories` API 5 zugewiesen.

Anfrage	Ausgewählte API	Erklärung
`https://api.example.com/orders`	`API 2`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v1/items`	`API 3`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v2/items`	`API 4`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat. Das `123` am Ende der Anforderung hat keinen Einfluss auf die Auswahl.

Anfrage	Ausgewählte API	Erklärung
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat.
`https://api.example.com/customers`	`API 1`	API Gateway verwendet das leere Mapping als Catch-All.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspräfix hat. Bei einem benutzerdefinierten Domain-Namen, der mit einstufigen Mappings konfiguriert ist, z. B. nur `https://api.example.com/orders` und `https://api.example.com/`, würde API Gateway `API 1` auswählen, da es keinen passenden Pfad mit `ordersandmore` gibt.

Einschränkungen

- Bei einer API-Zuordnung müssen sich der benutzerdefinierte Domainname und die zugewiesenen APIs im selben Konto befinden. AWS
- API-Mappings dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: `$-_.+!*'()/`.
- Die maximale Länge für den Pfad in eines API-Mappings beträgt 300 Zeichen.
- Es können 200 API-Zuweisungen mit mehreren Ebenen für jeden Domainnamen vorhanden sein.
- Sie können HTTP-APIs nur einem regionalen benutzerdefinierten Domain-Namen mit der TLS 1.2-Sicherheitsrichtlinie zuordnen.

- Sie können WebSocket APIs nicht demselben benutzerdefinierten Domainnamen zuordnen wie eine HTTP-API oder REST-API.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

AWS Serverless Application Model Vorlagen, die alle Ressourcen erstellen, finden Sie beispielsweise unter Sessions With SAM on GitHub.

AWS Management Console

So erstellen Sie ein API-Mapping

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie benutzerdefinierte Domain-Namen aus.
3. Wählen Sie einen benutzerdefinierten Domain-Namen aus, den Sie bereits erstellt haben.
4. Wählen Sie API-Mappings aus.
5. Wählen Sie API-Zuordnungen konfigurieren aus.
6. Wählen Sie Neue Zuordnung hinzufügen aus.
7. Geben Sie eine API, eine Phase und optional einen Pfad ein.
8. Wählen Sie Save (Speichern) aus.

AWS CLI

Mit dem folgenden AWS CLI Befehl wird eine API-Zuordnung erstellt. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1/orders` an die angegebene API und Phase.

> **Note**
>
> Um API-Mappings mit mehreren Ebenen zu erstellen, müssen Sie `apigatewayv2` verwenden.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

Das folgende AWS CloudFormation Beispiel erstellt eine API-Zuordnung.

> **Note**
>
> Um API-Mappings mit mehreren Ebenen zu erstellen, müssen Sie `AWS::ApiGatewayV2` verwenden.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Deaktivieren des Standardendpunkts für eine REST-API

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domänennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt. Clients können weiterhin eine Verbindung mit Ihrem Standardendpunkt herstellen, erhalten jedoch einen `403 Forbidden`-Statuscode.

> **Note**
>
> Wenn Sie den Standardendpunkt deaktivieren, wirkt sich dies auf alle Stufen einer API aus.

Der folgende AWS CLI Befehl deaktiviert den Standardendpunkt für eine REST-API.

```
aws apigateway update-rest-api \
    --rest-api-id abcdef123 \
    --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird.

Der folgende AWS CLI Befehl erstellt eine Bereitstellung.

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name dev
```

Benutzerdefinierte Zustandsprüfungen für das DNS-Failover konfigurieren

Sie können Amazon Route 53-Zustandsprüfungen verwenden, um den DNS-Failover von einer API-Gateway-API in einer primären AWS-Region zu einer in einer sekundären Region zu kontrollieren. Dies kann dazu beitragen, die Auswirkungen im Falle eines regionalen Problems zu mildern. Wenn Sie eine benutzerdefinierte Domäne verwenden, können Sie ein Failover durchführen, ohne dass die Clients die API-Endpunkte ändern müssen.

Wenn Sie Zielzustand bewerten für einen Aliasdatensatz auswählen, schlagen diese Datensätze nur fehl, wenn der API-Gateway-Service in der Region nicht verfügbar ist. In einigen Fällen kann es vor diesem Zeitpunkt zu Unterbrechungen bei Ihren eigenen API-Gateway-APIs kommen. Wenn Sie das DNS-Failover direkt steuern möchten, konfigurieren Sie benutzerdefinierte Route-53-Zustandsprüfungen für Ihre API-Gateway-APIs. In diesem Beispiel verwenden Sie einen CloudWatch Alarm, der den Betreibern hilft, das DNS-Failover zu kontrollieren. Weitere Beispiele und andere Überlegungen zur Konfiguration von Failover finden Sie unter Erstellen von Notfallwiederherstellungsmechanismen mithilfe von Route 53 und Durchführen von Route 53-Zustandsprüfungen für private Ressourcen in einer VPC mit AWS Lambda und. CloudWatch

Themen

- Voraussetzungen
- Schritt 1: Einrichten der Ressourcen
- Schritt 2: Initiieren des Failovers zur sekundären Region
- Schritt 3: Testen des Failovers
- Schritt 4: Rückkehr zur Primärregion

- Nächste Schritte: Anpassen und regelmäßig testen

Voraussetzungen

Um dieses Verfahren abzuschließen, müssen Sie die folgenden Ressourcen erstellen und konfigurieren:

- Eine Web-Domäne in Ihrem Besitz.
- Ein ACM-Zertifikat für diesen Domainnamen in zwei Teilen. AWS-Regionen Weitere Informationen finden Sie unter the section called “Vorbereiten von Zertifikaten in AWS Certificate Manager”.
- Eine gehostete Route-53-Zone für Ihren Domänennamen. Weitere Informationen finden Sie unter Arbeiten mit gehosteten Zonen im Entwicklerhandbuch für Amazon Route 53.

Weitere Informationen zum Erstellen von Route-53-Failover-DNS-Einträgen für die Domain-Namen finden Sie unter Eine Routing-Richtlinie auswählen im Amazon-Route-53-Entwicklerhandbuch. Weitere Informationen zur Überwachung eines CloudWatch Alarms finden Sie unter Überwachung eines CloudWatch Alarms im Amazon Route 53-Entwicklerhandbuch.

Schritt 1: Einrichten der Ressourcen

In diesem Beispiel erstellen Sie die folgenden Ressourcen, um das DNS-Failover für Ihren Domänennamen zu konfigurieren:

- API-Gateway-APIs in zwei AWS-Regionen
- API Gateway: benutzerdefinierte Domainnamen mit demselben Namen in zwei AWS-Regionen
- API-Gateway-API-Zuordnungen, die Ihre API-Gateway-APIs mit den benutzerdefinierten Domänennamen verbinden
- Route-53-Failover-DNS-Einträge für die Domänennamen
- Ein CloudWatch Alarm in der sekundären Region
- Eine Route 53-Zustandsprüfung auf der Grundlage des CloudWatch Alarms in der sekundären Region

Stellen Sie zunächst sicher, dass Sie über alle erforderlichen Ressourcen in den primären und sekundären Regionen verfügen. Die sekundäre Region sollte den Alarm und die Zustandsprüfung enthalten. So sind Sie nicht von der primären Region abhängig, um das Failover durchzuführen.

AWS CloudFormation Vorlagen, die diese Ressourcen erstellen, finden Sie beispielsweise unter primary.yamlund secondary.yaml.

> **Important**
>
> Stellen Sie vor dem Failover in die sekundäre Region sicher, dass alle erforderlichen Ressourcen verfügbar sind. Andernfalls ist Ihre API nicht für den Datenverkehr in der sekundären Region bereit.

Schritt 2: Initiieren des Failovers zur sekundären Region

Im folgenden Beispiel empfängt die Standby-Region eine CloudWatch Metrik und leitet ein Failover ein. Wir verwenden eine benutzerdefinierte Metrik, bei der das Eingreifen des Bedieners erforderlich ist, um das Failover einzuleiten.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 1 \
    --region us-west-1
```

Ersetzen Sie die Metrikdaten durch die entsprechenden Daten für den von Ihnen CloudWatch konfigurierten Alarm.

Schritt 3: Testen des Failovers

Rufen Sie Ihre API auf und vergewissern Sie sich, dass Sie eine Antwort von der sekundären Region erhalten. Wenn Sie die Beispielvorlagen in Schritt 1 verwendet haben, ändert sich die Antwort von „`{"message": "Hello from the primary Region!"}`“ nach dem Failover zu „`{"message": "Hello from the secondary Region!"}`“.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

Schritt 4: Rückkehr zur Primärregion

Um zur primären Region zurückzukehren, senden Sie eine CloudWatch Metrik, die dafür sorgt, dass die Integritätsprüfung bestanden wird.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 0 \
    --region us-west-1
```

Ersetzen Sie die Metrikdaten durch die entsprechenden Daten für den von Ihnen konfigurierten CloudWatch Alarm.

Rufen Sie Ihre API auf und vergewissern Sie sich, dass Sie eine Antwort von der primären Region erhalten. Wenn Sie die Beispielvorlagen in Schritt 1 verwendet haben, ändert sich die Antwort von „`{"message": "Hello from the secondary Region!"}`“ zu „`{"message": "Hello from the primary Region!"}`“.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Nächste Schritte: Anpassen und regelmäßig testen

Dieses Beispiel zeigt eine Möglichkeit, das DNS-Failover zu konfigurieren. Sie können eine Vielzahl von CloudWatch Metriken oder HTTP-Endpunkten für die Integritätsprüfungen zur Verwaltung des Failovers verwenden. Testen Sie Ihre Failover-Mechanismen regelmäßig, um sicherzustellen, dass sie erwartungsgemäß funktionieren und dass die Bediener mit Ihren Failover-Verfahren vertraut sind.

Leistung der REST-APIs optimieren

Nachdem Sie Ihre API für den Aufruf zur Verfügung gestellt haben, stellen Sie vielleicht fest, dass sie optimiert werden muss, um die Reaktionsfähigkeit zu verbessern. API Gateway bietet einige Strategien zur Optimierung Ihrer API, wie Antwort-Caching und Payload-Komprimierung. In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen aktivieren.

Themen

Aktivieren von API-Caching für verbesserte Reaktionsfähigkeit

Sie können API-Caching in Amazon API Gateway aktivieren, um die Antworten Ihres Endpunkts zu cachen. Mit Caching können Sie die Anzahl der an Ihren Endpunkt gemachten Aufrufe verringern und die Latenz der Anforderungen an Ihre API verbessern.

Wenn Sie das Caching für eine Phase aktivieren, speichert API Gateway Antworten von Ihrem Endpunkt für einen bestimmten Zeitraum time-to-live (TTL) in Sekunden zwischen. Anschließend beantwortet API Gateway die Anfrage, indem es die Endpunkt-Antwort aus dem Cache ausliest, anstatt eine Anfrage an Ihren Endpunkt zu senden. Der Standard-TTL-Wert für API-Caching ist 300 Sekunden. Die maximale TTL-Wert ist 3.600 Sekunden. TTL = 0 bedeutet, dass die Zwischenspeicherung deaktiviert ist.

> **Note**
>
> Caching ist die sinnvollste Tätigkeit. Sie können die `CacheMissCount` Metriken `CacheHitCount` und in Amazon verwenden CloudWatch , um Anfragen zu überwachen, die API Gateway aus dem API-Cache bedient.

Die maximale Größe einer Antwort, die zwischengespeichert werden kann, ist 1 048 576 Bytes. Die Cache-Datenverschlüsselung kann die Größe der Antwort erhöhen, wenn sie zwischengespeichert wird.

Dies ist ein HIPAA-berechtigter Service. Weitere Informationen AWS zum US-amerikanischen Health Insurance Portability and Accountability Act von 1996 (HIPAA) und zur Nutzung von AWS Diensten zur Verarbeitung, Speicherung und Übertragung geschützter Gesundheitsinformationen (PHI) finden Sie unter HIPAA Overview.

> **Important**
>
> Wenn Sie das Caching für eine Stufe aktivieren, wird standardmäßig nur die Zwischenspeicherung von `GET`-Methoden aktiviert. Dies hilft, die Sicherheit und die Verfügbarkeit Ihrer API zu gewährleisten. Sie können das Caching für andere Methoden aktivieren, indem Sie die Methodeneinstellungen überschreiben.

> **Important**
>
> Caching wird basierend auf der von Ihnen ausgewählten Cache-Größe nach Stunden berechnet. Caching kommt nicht für das kostenlose Kontingent in Frage. AWS Weitere Informationen finden Sie unter API-Gateway-Preise.

Caching von Amazon API Gateway aktivieren

In API Gateway können Sie das Caching für eine bestimmte Phase aktivieren.

Wenn Sie Caching aktivieren, müssen Sie eine Cache-Kapazität auswählen. Im Allgemeinen bietet eine größere Kapazität eine bessere Leistung, kostet aber auch mehr. Informationen zu den unterstützten Cachegrößen finden Sie cacheClusterSizein der API Gateway API Reference.

API Gateway ermöglicht das Caching durch die Erstellung einer dedizierten Cache-Instance. Dieser Vorgang kann bis zu 4 Minuten dauern.

API Gateway ändert die Caching-Kapazität durch Löschen der vorhandenen Cache-Instance und Erstellen einen neuen mit einer geänderten Kapazität. Alle Daten im Cache werden gelöscht.

> **Note**
>
> Die Cache-Kapazität wirkt sich auf die CPU- und Netzwerkbandbreite der Cache-Instance aus. Infolgedessen kann sich die Cache-Kapazität auf die Performance Ihres Caches auswirken.
> API Gateway empfiehlt, dass Sie einen 10-Minuten-Belastungstest durchführen, um zu überprüfen, ob Ihre Cache-Kapazität für Ihre Arbeitslast geeignet ist. Stellen Sie sicher, dass der Verkehr während des Lasttests den Produktionsverkehr widerspiegelt. Dazu gehören zum Beispiel Ramp Up, konstanter Verkehr und Verkehrsspitzen. Der Lasttest sollte Antworten enthalten, die aus dem Cache bereitgestellt werden können, sowie eindeutige Antworten, die dem Cache Elemente hinzufügen. Überwachen Sie die Latenz-, 4xx-, 5xx-, Cache-Hit- und Cache-Fehl-Metriken während des Belastungstests. Passen Sie Ihre Cache-Kapazität basierend auf diesen Metriken nach Bedarf an. Weitere Informationen zu Lasttests finden Sie unter Wie wähle ich die beste Kapazität für den API-Gateway-Cache aus, um zu vermeiden, dass ein Ratenlimit erreicht wird?.

In der API Gateway Gateway-Konsole konfigurieren Sie das Caching auf der Seite Stages. Sie stellen den Stage-Cache bereit und geben eine Standardeinstellung für den Cache auf Methodenebene an. Wenn Sie den Standardcache auf Methodenebene aktivieren, wird das Caching auf Methodenebene für alle `GET` Methoden auf Ihrer Stufe aktiviert, sofern diese Methode nicht über eine Methodenüberschreibung verfügt. Alle zusätzlichen `GET` Methoden, die Sie in Ihrer Phase bereitstellen, verfügen über einen Cache auf Methodenebene. Um die Caching-Einstellungen auf Methodenebene für bestimmte Methoden Ihrer Phase zu konfigurieren, können Sie Methodenüberschreibungen verwenden. Weitere Informationen zu Methodenüberschreibungen finden Sie unter. the section called “Überschreiben des Stufen-Caching für das Methoden-Caching”

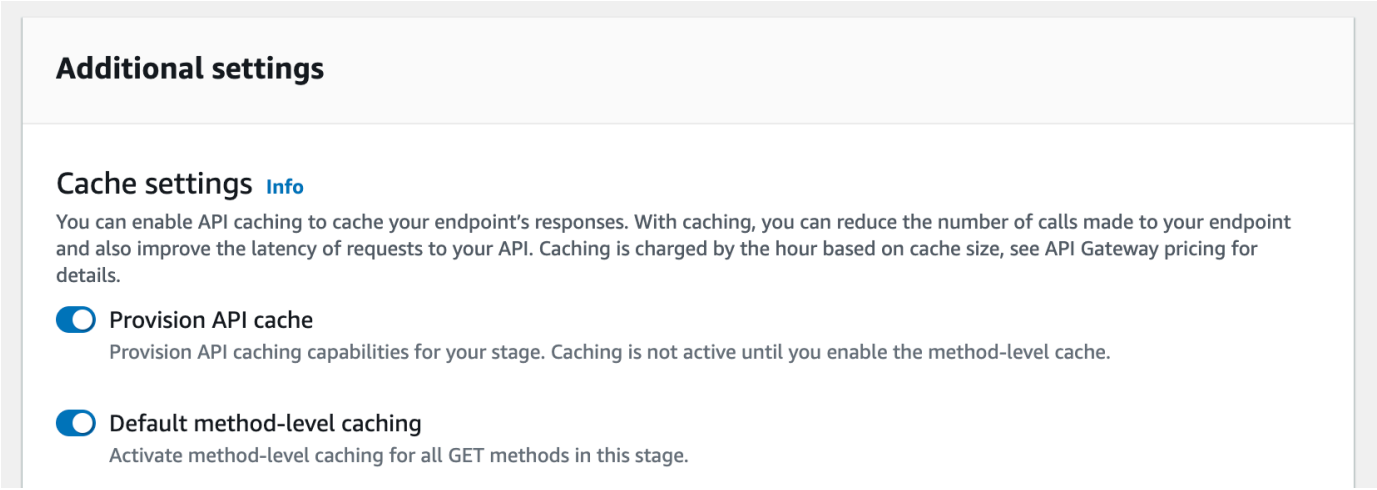
So konfigurieren Sie API-Caching für eine bestimmte Stufe:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Stages.
3. Wählen Sie in der API-Liste Stages den Namen der Stufe aus.
4. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
5. Aktivieren Sie unter Zusätzliche Einstellungen für Cache-Einstellungen die Option API-Cache bereitstellen.

   Dadurch wird ein Cache-Cluster für Ihre Phase bereitgestellt.
6. Um das Caching für Ihre Phase zu aktivieren, aktivieren Sie das Standard-Caching auf Methodenebene.

   Dadurch wird das Caching auf Methodenebene für alle Methoden in Ihrer Phase aktiviert. `GET` Alle zusätzlichen `GET` Methoden, die Sie in dieser Phase bereitstellen, verfügen über einen Cache auf Methodenebene.

   > **Note**
   >
   > Wenn Sie bereits über eine Einstellung für einen Cache auf Methodenebene verfügen, wirkt sich eine Änderung der Standardeinstellung für das Zwischenspeichern auf Methodenebene nicht auf diese vorhandene Einstellung aus.



7. Wählen Sie Änderungen speichern aus.

> **Note**
>
> Das API Gateway benötigt etwa 4 Minuten für das Erstellen und Löschen eines Caches. Wenn ein Cache erstellt wird, ändert sich der Wert des Cache-Clusters von auf. `Create in progress` `Active` Wenn das Löschen des Cache abgeschlossen ist, ändert sich der Wert des Cache-Clusters von `Delete in progress` auf`Inactive`.
> Wenn Sie das Caching auf Methodenebene für alle Methoden auf Ihrer Stufe aktivieren, ändert sich der Standardwert für das Zwischenspeichern auf Methodenebene auf. `Active`
> Wenn Sie das Caching auf Methodenebene für alle Methoden in Ihrer Phase deaktivieren, ändert sich der Standardwert für das Zwischenspeichern auf Methodenebene auf. `Inactive`
> Wenn Sie bereits über eine Einstellung für einen Cache auf Methodenebene verfügen, wirkt sich eine Änderung des Cache-Status nicht auf diese Einstellung aus.

Wenn Sie das Caching in den Cache-Einstellungen einer Stufe aktivieren, werden nur `GET`-Methoden zwischengespeichert. Um die Sicherheit und Verfügbarkeit Ihrer API zu gewährleisten, empfehlen wir Ihnen, diese Einstellung nicht zu ändern. Sie können das Caching aber auch für andere Methoden aktivieren, indem Sie die Methodeneinstellungen überschreiben.

Wenn Sie sich vergewissern möchten, ob die Zwischenspeicherung wie erwartet funktioniert, haben Sie zwei allgemeine Optionen:

- Untersuchen Sie die CloudWatch Metriken von CacheHitCountund CacheMissCountfür Ihre API und Phase.
- Setzen Sie einen Zeitstempel in der Antwort.

> **Note**
>
> Sie sollten den `X-Cache` Header aus der CloudFront Antwort nicht verwenden, um festzustellen, ob Ihre API von Ihrer API Gateway Gateway-Cache-Instance bedient wird.

Überschreiben Sie API Gateway API-Gateway-Caching auf Stufenebene für Caching auf Methodenebene

Sie können die Cache-Einstellungen auf Stufenebene überschreiben, indem Sie das Caching für eine bestimmte Methode ein- oder ausschalten. Sie können auch den TTL-Zeitraum ändern oder die Verschlüsselung für zwischengespeicherte Antworten ein- oder ausschalten.

Wenn Sie die Standardeinstellung für das Zwischenspeichern auf Methodenebene in den Details der Phase ändern, hat dies keine Auswirkungen auf die Cache-Einstellungen auf Methodenebene, für die es Überschreibungen gibt.

Wenn Sie davon ausgehen, dass eine zwischengespeicherte Methode sensible Daten in ihren Antworten erhält, wählen Sie in den Cache-Einstellungen die Option Encrypt cache data aus.

So konfigurieren Sie API-Caching für einzelne Methoden mithilfe der Konsole:

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie die API.
3. Wählen Sie Stages.
4. Erweitern Sie in der Liste Stufen für die API die Stufe, und wählen Sie eine Methode in der API aus.
5. Wählen Sie im Abschnitt Methodenüberschreibungen die Option Bearbeiten aus.
6. Aktivieren oder deaktivieren Sie im Abschnitt Methodeneinstellungen die Option Methoden-Cache aktivieren oder deaktivieren Sie die gewünschten Optionen.

> **Note**
>
> Das Caching ist erst aktiv, wenn Sie einen Cache-Cluster für Ihre Phase bereitstellen.

7. Wählen Sie Speichern.

Verwenden der Methoden-/Integrationsparameter als Cache-Schlüssel, um zwischengespeicherte Antworten zu indizieren

Wenn eine zwischengespeicherte Methode oder Integration Parameter hat, z. B. in Form von kundenspezifischen Headers, URL-Pfaden oder Abfrage-Zeichenfolgen, können Sie einige oder alle der Parameter verwenden, um Cache-Schlüssel zu bilden. API Gateway kann die Antworten der Methode in Abhängigkeit von den verwendeten Parameterwerten zwischenspeichern.

> **Note**
>
> Cache-Schlüssel sind beim Einrichten von Caching für eine Ressource erforderlich.

Angenommen, Sie haben eine Anforderung im folgenden Format:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

In dieser Anforderung kann `type` den Wert `admin` oder `regular` annehmen. Wenn Sie den `type`-Parameter als Teil des Cache-Schlüssels einbinden, werden die Antworten aus `GET /users?type=admin` getrennt von denen aus `GET /users?type=regular` zwischengespeichert.

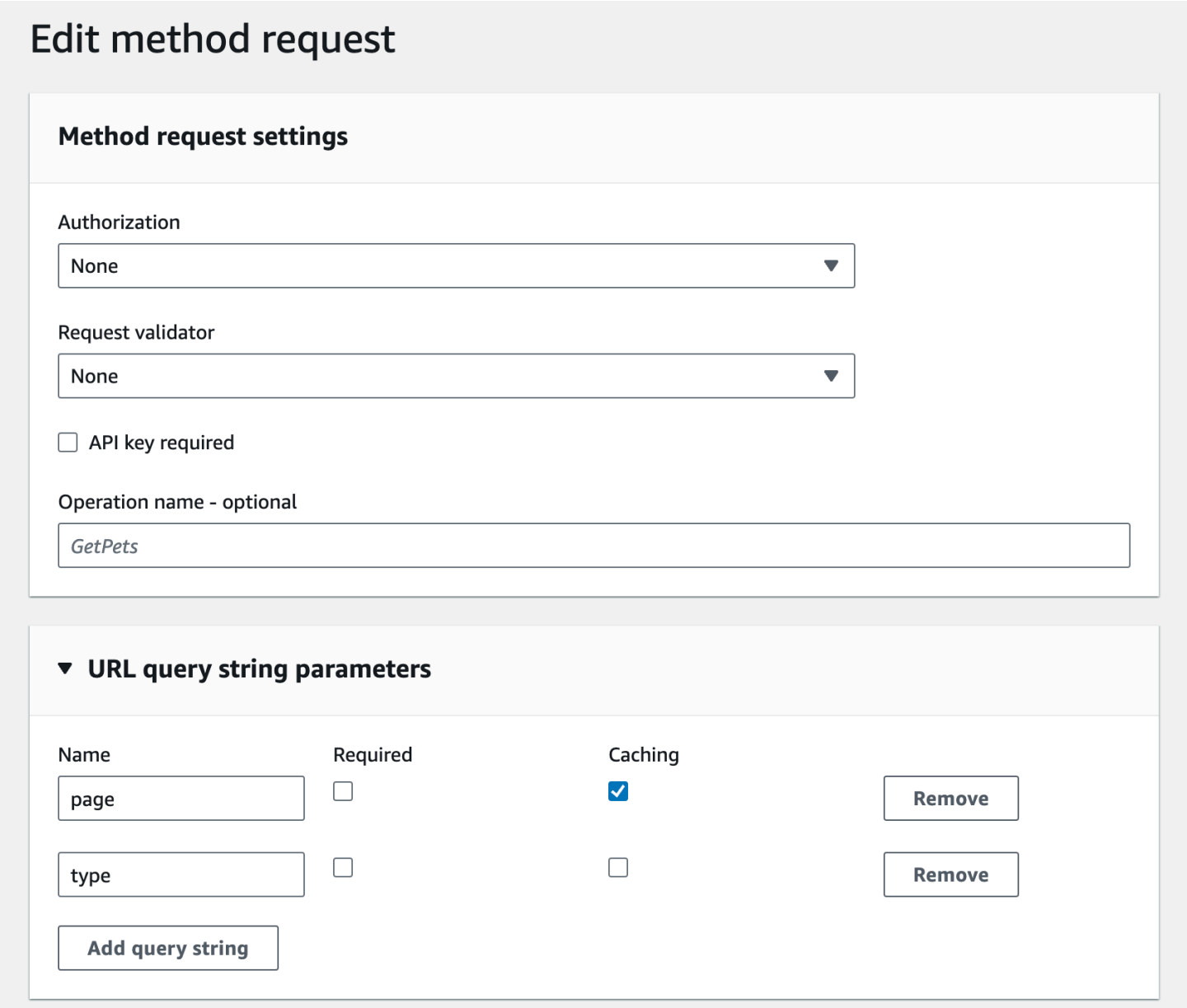
Wenn eine Methode oder Integrationsanforderung mehr als einen Parameter annimmt, können Sie einige oder alle dieser Parameter einschließen, um den Cache-Schlüssel zu erstellen. Beispielsweise können Sie für die folgende Anforderung nur den `type`-Parameter in den Cache-Schlüssel einschließen, in der aufgelisteten Reihenfolge innerhalb eines TTL-Zeitraums:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

Die Antwort von dieser Anforderung wird im Cache gespeichert und verwendet, um die folgende Anforderung zu bedienen:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Um einen Methoden- oder Integrationsanforderungsparameter als Teil eines Cache-Schlüssels in die API Gateway-Konsole aufzunehmen, wählen Sie Caching, nachdem Sie den Parameter hinzugefügt haben.



API-Stufen-Cache in API Gateway leeren

Wenn API-Zwischenspeicherung aktiviert ist, können Sie den Cache Ihrer API-Stufe löschen, um sicherzustellen, dass die Clients Ihrer API die neuesten Antworten von Ihren Integrationsendpunkten erhalten.

Wenn Sie den API-Stufen-Cache leeren möchten, wählen Sie das Menü Stufenaktionen und dann Stufen-Cache leeren aus.

> **Note**
>
> Nachdem der Cache gelöscht wurde, werden Antworten aus dem Integrationsendpunkt bedient, bis der Cache wieder aufgebaut ist. Während dieses Zeitraums kann sich die Anzahl der Anforderungen an den Integrationsendpunkt erhöhen. Dadurch kann sich die Gesamt-Latenz Ihrer API zeitweilig erhöhen.

API Gateway-Cache-Eintrag ungültig machen

Ein Client Ihrer API kann einen vorhandenen Cache-Eintrag entwerten und ihn vom Integrationsendpunkt für einzelne Anforderungen neu laden. Der Client muss eine Anforderung senden, die den Header `Cache-Control: max-age=0` enthält. Der Client erhält die Antwort direkt vom Integrationsendpunkt anstelle des Zwischenspeichers, wenn der Client hierfür eine entsprechende Berechtigung hat. Dies ersetzt den vorhandenen Cache-Eintrag durch die neue Antwort, die vom Integrationsendpunkt abgerufen wird.

Um einem Aufrufer eine Berechtigung zu erteilen, fügen Sie an die IAM-Ausführungsrolle für den Client eine Richtlinie im folgenden Format an.

> **Note**
>
> Die kontoübergreifende Cache-Invalidierung wird nicht unterstützt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"
      ]
    }
  ]
```

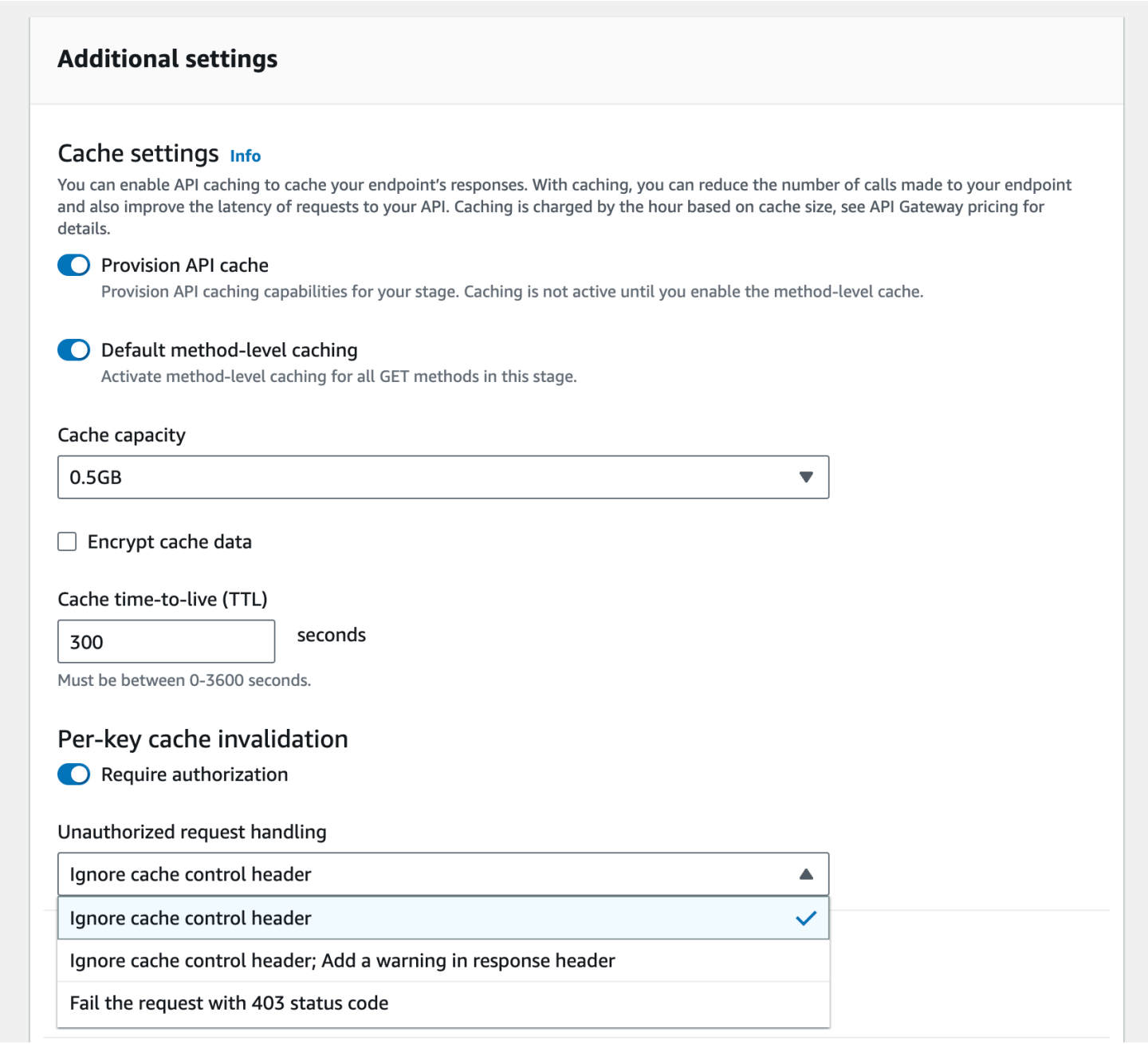
```
}
```

Diese Richtlinie erlaubt es dem API Gateway-Ausführungsdienst, den Cache für Anfragen zur angegebenen Ressource (oder den angegebenen Ressourcen) ungültig zu machen. Um eine Gruppe zum Ziel gesetzter Ressourcen anzugeben, verwenden Sie ein Platzhalterzeichen (*) für `account-id`, `api-id` und andere Angaben im ARN-Wert von `Resource`. Weitere Informationen über das Festlegen von Berechtigungen für den API Gateway-Ausführungsdienst finden Sie unter Kontrollieren des Zugriffs auf eine API mit IAM-Berechtigungen.

Wenn sie keine `InvalidateCache`-Richtlinie angeben (oder das Kontrollkästchen Autorisierung erforderlich in der Konsole aktivieren), kann jeder beliebige Client den API-Zwischenspeicher entwerten. Wenn die meisten oder alle Clients den API-Cache entwerten, kann dies zu einer erheblichen Steigerung der Latenz Ihrer API führen.

Wenn die Richtlinie eingerichtet ist, wird das Caching aktiviert und eine Autorisierung ist erforderlich.

Sie können steuern, wie nicht autorisierte Anfragen behandelt werden, indem Sie in der API Gateway Gateway-Konsole eine Option unter Behandlung nicht autorisierter Anfragen auswählen.



Die drei Optionen haben folgende Verhaltensweisen zur Folge:

- Fail the request with 403 status code: gibt die Antwort „403 Unauthorized“ zurück.

  Geben Sie zum Einstellen dieser Option über die API ei `FAIL_WITH_403`.
- Ignore cache control header Add a warning in response header: Bearbeiten der Anfrage und Hinzufügen eines Warn-Headers in der Antwort.

  Geben Sie zum Einstellen dieser Option über die API ei `SUCCEED_WITH_RESPONSE_HEADER`.

- Ignore cache control header: Bearbeiten der Anfrage ohne Hinzufügen eines Warn-Headers in der Antwort.

  Geben Sie zum Einstellen dieser Option über die API ei `SUCCEED_WITHOUT_RESPONSE_HEADER`.

Aktivieren der Nutzlastkomprimierung für eine API

API Gateway ermöglicht es Ihrem Client, über eine der unterstützten Inhaltskodierungen Ihre API mit komprimierten Payloads aufzurufen. API Gateway unterstützt standardmäßig die Dekomprimierung der Methodenanforderungsnutzlast. Sie müssen Ihre API allerdings so konfigurieren, dass die Komprimierung der Methodenantwortnutzlast aktiviert wird.

Zum Aktivieren der Komprimierung für eine `API` legen Sie für die Eigenschaft `minimumCompressionsSize` beim oder nach dem Erstellen der API eine nicht negative Ganzzahl zwischen 0 und 10485760 (10 M Bytes) fest. Zum Deaktivieren der Komprimierung in der API legen Sie `minimumCompressionSize` auf Null fest oder entfernen Sie die Eigenschaft. Sie können die Komprimierung für eine API aktivieren oder deaktivieren, indem Sie die API Gateway Gateway-Konsole AWS CLI, die oder die API Gateway Gateway-REST-API verwenden.

Wenn Sie die Komprimierung auf eine Nutzlast jeder beliebigen Größe angewendet werden soll, legen Sie den Wert `minimumCompressionSize` auf Null fest. Durch die Komprimierung einer kleinen Datengröße kann sich jedoch die endgültige Datengröße erhöhen. Darüber hinaus kann die Komprimierung in API Gateway und die Dekomprimierung im Client die Gesamtlatenzzeit erhöhen und mehr Rechenzeit erfordern. Führen Sie Testläufe für Ihre API aus, um den optimalen Wert zu bestimmen.

Der Client kann eine API-Anfrage mit einer komprimierten Nutzlast und einem geeigneten `Content-Encoding`-Header für API Gateway senden, um geeignete Zuweisungsvorlagen zu komprimieren und anzuwenden, bevor die Anfrage dem Integrationsendpunkt übergeben wird. Nachdem die Komprimierung aktiviert und die API bereitgestellt wurde, kann der Client eine API-Antwort mit einer komprimierten Nutzlast empfangen, wenn ein geeigneter `Accept-Encoding`-Header in der Methodenanforderung angegeben ist.

Wenn der Integrationsendpunkt nicht komprimierte JSON-Nutzlasten erwartet und zurückgibt, ist jede Zuweisungsvorlage, die für eine nicht komprimierte JSON-Nutzlast konfiguriert ist, für die komprimierte Nutzlast anwendbar. Für eine komprimierte Methodenanforderungsnutzlast dekomprimiert API Gateway die Nutzlast, wendet die Zuweisungsvorlage an und übergibt dem Integrationsendpunkt die zugewiesene Anfrage. Für eine nicht komprimierte

Methodenanforderungsnutzlast wendet API Gateway die Zuweisungsvorlage an, komprimiert die zugewiesene Nutzlast und gibt die komprimierte Nutzlast an den Client zurück.

Themen

- Aktivieren der Nutzlastkomprimierung für eine API
- Aufrufen einer API-Methode mit einer komprimierten Nutzlast
- Empfangen einer API-Antwort mit einer komprimierten Nutzlast

Aktivieren der Nutzlastkomprimierung für eine API

Sie können die Komprimierung für eine API mithilfe der API Gateway Gateway-Konsole AWS CLI, des oder eines AWS SDK aktivieren.

Im Fall einer vorhandenen API müssen Sie nach der Aktivierung der Komprimierung die API bereitstellen, damit die Änderungen wirksam werden. Im Fall einer neuen API können Sie die API nach Abschluss der API-Einrichtung bereitstellen.

> **Note**
>
> Die Inhaltskodierung mit der höchsten Priorität muss von API Gateway unterstützt werden. Wenn dies nicht der Fall ist, wird keine Komprimierung auf die Antwortnutzlast angewendet.

Themen

- Payload-Komprimierung für eine API über die API Gateway-Konsole aktivieren
- Aktivieren Sie die Payload-Komprimierung für eine API mit AWS CLI
- Von API Gateway unterstützte Inhaltskodierungen

Payload-Komprimierung für eine API über die API Gateway-Konsole aktivieren

Im folgenden Verfahren wird beschrieben, wie Sie die Nutzlastkomprimierung für eine API aktivieren.

So aktivieren Sie die Payload-Komprimierung über die API Gateway-Konsole:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.

3. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Aktivieren Sie die Inhaltskodierung, um die Nutzlast-Komprimierung zu aktivieren. Geben Sie einen Wert für die Mindestkomprimierungsgröße (in Byte) unter Mindesttextgröße ein. Um die Komprimierung zu deaktivieren, schalten Sie die Option Inhaltskodierung aus.
6. Wählen Sie Änderungen speichern aus.

Aktivieren Sie die Payload-Komprimierung für eine API mit AWS CLI

Rufen Sie den `create-rest-api`Befehl wie folgt auf, AWS CLI um eine neue API zu erstellen und die Komprimierung zu aktivieren:

```
aws apigateway create-rest-api \
    --name "My test API" \
    --minimum-compression-size 0
```

Rufen Sie den `update-rest-api`Befehl wie folgt auf, AWS CLI um die Komprimierung für eine bestehende API zu aktivieren:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

Die Eigenschaft `minimumCompressionSize` weist eine nicht negative ganze Zahl zwischen 0 und 10485760 (10 M Bytes) auf. Mit ihr wird die Komprimierung gemessen. Wenn die Nutzlastgröße kleiner als dieser Wert ist, wird die Komprimierung oder Dekomprimierung nicht auf die Nutzlast angewendet. Wenn er auf Null festgelegt ist, ist die Komprimierung für jede beliebige Nutzlastgröße erlaubt.

Rufen Sie den `update-rest-api`Befehl wie folgt auf, AWS CLI um die Komprimierung zu deaktivieren:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=
```

Sie können `value` auch auf eine leere Zeichenfolge `""` festlegen oder die Eigenschaft `value` im vorhergehenden Aufruf vollständig weglassen.

Von API Gateway unterstützte Inhaltskodierungen

API Gateway unterstützt die folgenden Inhaltskodierungen:

- `deflate`
- `gzip`
- `identity`

API Gateway unterstützt auch das folgende `Accept-Encoding` Header-Format gemäß der Spezifikation RFC 7231:

- `Accept-Encoding:deflate,gzip`
- `Accept-Encoding:`
- `Accept-Encoding:*`
- `Accept-Encoding:deflate;q=0.5,gzip;q=1.0`
- `Accept-Encoding:gzip;q=1.0,identity;q=0.5,*;q=0`

Aufrufen einer API-Methode mit einer komprimierten Nutzlast

Um eine API-Anfrage mit einer komprimierten Nutzlast zu erstellen, muss der Client den `Content-Encoding`-Header mit einer der unterstützten Inhaltscodierungen festlegen.

Angenommen, Sie sind ein API-Client und möchten die PetStore API-Methode (`POST /pets`) aufrufen. Rufen Sie die Methode nicht mit der folgenden JSON-Ausgabe auf:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

Stattdessen können Sie die Methode mit der gleichen Nutzlast aufrufen, die mit der GZIP-Codierung komprimiert wird:

```
POST /pets
```

```
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

[binary gzip data]
```

Wenn API Gateway die Anfrage erhält, prüft es, ob die angegebene Inhaltskodierung unterstützt wird. Anschließend wird versucht, die Nutzlast mit der angegebenen Inhaltscodierung zu dekomprimieren. Wenn die Dekomprimierung erfolgreich ist, wird die Anforderung an den Integrationsendpunkt gesendet. Wenn die angegebene Kodierung nicht unterstützt wird oder der gelieferte Payload nicht mit der angegebenen Kodierung komprimiert ist, gibt API Gateway die `415 Unsupported Media Type`-Fehlerantwort zurück. Der Fehler wird nicht in CloudWatch Logs protokolliert, wenn er in der frühen Phase der Dekomprimierung auftritt, bevor Ihre API und Ihre Phase identifiziert wurden.

Empfangen einer API-Antwort mit einer komprimierten Nutzlast

Beim Erstellen einer Anforderung in einer komprimierungsfähigen API kann der Client festlegen, dass er eine komprimierte Antwortnutzlast eines bestimmten Formats erhält, indem er einen `Accept-Encoding`-Header mit einer unterstützten Inhaltscodierung angibt.

API Gateway komprimiert den Antwortpayload nur, wenn die folgenden Bedingungen erfüllt sind:

- Der `Accept-Encoding`-Header der eingehenden Anforderung weist eine unterstützte Inhaltscodierung und ein entsprechendes Format auf.

  > **Note**
  >
  > Wenn der Header nicht festgelegt ist, ist der Standardwert * gemäß RFC 7231. In einem solchen Fall komprimiert API Gateway den Payload nicht. Einige Browser oder Clients fügen komprimierungsfähigen Anforderungen möglicherweise automatisch `Accept-Encoding` (z. B. `Accept-Encoding:gzip, deflate, br`) hinzu. Dies kann die Komprimierung des Payloads in API Gateway auslösen. Ohne explizite Angabe der unterstützten `Accept-Encoding`-Header-Werte komprimiert API Gateway den Payload nicht.

- Die `minimumCompressionSize` wird in der API festgelegt, um die Komprimierung zu aktivieren.
- Die Integrationsantwort enthält keinen `Content-Encoding`-Header.
- Die Größe einer Integrationsantwortnutzlast nach Anwendung der geltenden Zuweisungsvorlage ist größer als oder gleich dem angegebenen `minimumCompressionSize`-Wert.

API Gateway wendet ggf. eine Zuweisungsvorlagen an, die für die Integrationsantwort konfiguriert ist, bevor die Nutzlast komprimiert wird. Wenn die Integrationsantwort einen `Content-Encoding`-Header enthält, geht API Gateway davon aus, dass die Integrationsantwortnutzlast bereits komprimiert ist und überspringt die Komprimierungsverarbeitung.

Ein Beispiel ist das PetStore API-Beispiel und die folgende Anfrage:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

Das Backend beantwortet die Anforderung mit einer nicht komprimierten JSON-Nutzlast ähnlich wie der folgenden:

```
200 OK

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

Zum Empfangen dieser Ausgabe als komprimierte Nutzlast kann Ihr API-Client eine Anforderung wie folgt senden:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Der Client erhält die Antwort mit einem `Content-Encoding`-Header und einer GZIP-kodierten Nutzlast ähnlich wie den folgenden:

```
200 OK
Content-Encoding:gzip
...

���RP�

J�)JV
�:P^IeA*������+(�L ��X�YZ�ku0L0B7!9��C#�&����Y��a���^�X
```

Wird die Antwortnutzlast komprimiert, wird nur die komprimierte Datengröße für die Datenübertragung in Rechnung gestellt.

Verteilen Ihrer REST-API an Kunden

Dieser Abschnitt enthält Details zur Verteilung Ihrer API-Gateway-APIs an Ihre Kunden. Die Verteilung Ihrer API umfasst die Generierung von SDKs, die Ihre Kunden herunterladen und in ihre Client-Anwendungen integrieren können, die Dokumentation Ihrer API, damit Kunden wissen, wie sie von ihren Client-Anwendungen aus aufgerufen werden kann und die Bereitstellung Ihrer API als Teil von Produktangeboten.

Themen

- Erstellen und Verwenden von Nutzungsplänen mit API-Schlüsseln
- Dokumentieren von REST-APIs
- Generieren eines SDK für eine REST-API in API Gateway
- Verkaufen Sie Ihre API Gateway Gateway-APIs über AWS Marketplace

Erstellen und Verwenden von Nutzungsplänen mit API-Schlüsseln

Nachdem Sie die APIs erstellt, getestet und bereitgestellt haben, können Sie API Gateway-Nutzungspläne verwenden, um die APIs als Produktangebot für Ihre Kunden bereitzustellen. Sie können Nutzungspläne und API-Schlüssel konfigurieren, um Kunden den Zugriff auf ausgewählte APIs zu ermöglichen, und damit beginnen, Anfragen an diese APIs basierend auf definierten Limits und Kontingenten zu drosseln. Diese können auf API- oder API-Methodenebene festgelegt werden.

Was sind Nutzungspläne und API-Schlüssel?

Ein Nutzungsplan gibt an, wer Zugriff auf eine oder mehrere bereitgestellte API-Stufen und Methoden hat. Es wird optional die Zielanforderungsrate festgelegt, um die Drosselungsrate der Anfragen zu starten. Der Plan verwendet API-Schlüssel zur Identifizierung von API-Clients und derjenigen, welche auf die zugehörigen API-Stufen jedes Schlüssels zugreifen können.

API-Schlüssel sind alphanumerische Zeichenfolgenwerte, mit denen Sie Anwendungsentwicklerkunden Zugriff auf Ihre API gewähren. Sie können API-Schlüssel zusammen mit Lambda-Genehmigern, IAM-Rollen oder Amazon Cognito verwenden, um den Zugriff auf Ihre APIs zu steuern. API Gateway kann API-Schlüssel für Sie erstellen. Alternativ können Sie Schlüssel aus einer CSV-Datei importieren. Sie können einen API-Schlüssel in API Gateway erstellen oder ihn aus einer externen Quelle in API Gateway importieren. Weitere Informationen finden Sie unter the section called “Einrichten von API-Schlüsseln mit der API-Gateway-Konsole”.

Ein API-Schlüssel verfügt über einen Namen und einen Wert. (Die Begriffe „API-Schlüssel” und „API-Schlüsselwert” werden häufig synonym verwendet.) Namen dürfen nicht mehr als 1024 Zeichen enthalten. Der Wert ist eine alphanumerische Zeichenfolge mit 20 bis 128 Zeichen, beispielsweise, `apikey1234abcdefghij0123456789`.

> **Important**
>
> API-Schlüsselwerte müssen eindeutig sein. Wenn Sie versuchen, zwei API-Schlüssel mit unterschiedlichen Namen und demselben Wert zu erstellen, hält API Gateway sie für ein- und denselben API-Schlüssel.
> Ein API-Schlüssel kann mehreren Nutzungsplänen zugeordnet werden. Ein Nutzungsplan kann mehreren Stufen zugeordnet werden. Ein bestimmter API-Schlüssel kann jedoch nur einem Nutzungsplan für jede Stufe Ihres API zugeordnet werden.

Ein -Drosselungslimits legt den Zielpunkt fest, an dem die Anforderungsdrosselung beginnen soll. Dies kann auf API- oder API-Methodenebene festgelegt werden.

Ein Kontingentlimit legt die maximale Zielanzahl der Anforderungen fest, die mit einem bestimmten API-Schlüssel innerhalb eines vorgegebenen Zeitraums übermittelt werden können. Sie können einzelnen API-Methoden so konfigurieren, dass der API-Schlüssel autorisiert werden muss (abhängig von der Konfiguration des Nutzungsplans).

Die Drosselungs- und Kontingentlimits gelten für Anforderungen einzelner API-Schlüssel, die über alle API-Stufen eines Nutzungsplans aggregiert werden.

> **Note**
>
> Die -Drosselungsrate und die Kontingente der Nutzungspläne sind keine harten Grenzen und werden Best-Effort-Basis angewendet. In einigen Fällen können Kunden die von Ihnen festgelegten Kontingente überschreiten. Verlassen Sie sich nicht auf Nutzungsplankontingente oder Drosseln, um Kosten zu kontrollieren oder den Zugriff auf eine API zu blockieren. Überlegen Sie sich, AWS Budgets zu verwenden, um Kosten zu überwachen und AWS WAF, um API-Anfragen zu verwalten.

Bewährte Methoden für API-Schlüssel und Nutzungspläne

Im Folgenden finden Sie empfohlene bewährte Methoden für die Verwendung von API-Schlüsseln und Nutzungsplänen.

> **Important**
>
> - Verwenden Sie keine API-Schlüssel für die Authentifizierung oder Autorisierung der Zugriffskontrolle für Ihre APIs. Zum Einen kann, falls sich mehrere APIs in einem Nutzungsplan befinden, ein Benutzer mit einem gültigen API-Schlüssel für eine API in diesem Nutzungsplan auf alle APIs in diesem Nutzungsplan zugreifen. Verwenden Sie für die Zugriffskontrolle auf Ihre API stattdessen eine IAM-Rolle, einen Lambda-Genehmiger oder einen Amazon Cognito-Benutzerpool.
> - Verwenden Sie API-Schlüssel, die API Gateway generiert. API-Schlüssel sollten keine vertraulichen Informationen enthalten. Clients übertragen sie normalerweise in Headern, die protokolliert werden können.

- Wenn Sie ein Entwicklerportal verwenden, um Ihre APIs zu veröffentlichen, beachten Sie, dass alle Ihre APIs in einem bestimmten Nutzungsplan von Kunden abonniert werden können, auch wenn Sie sie für Ihre Kunden nicht sichtbar gemacht haben.
- In einigen Fällen können Kunden die von Ihnen festgelegten Kontingente überschreiten. Verlassen Sie sich nicht auf Nutzungspläne, um die Kosten zu kontrollieren. Ziehen Sie es in Betracht, AWS Budgets zu verwenden, um Kosten zu überwachen und AWS WAF, um API-Anfragen zu verwalten.

- Nachdem Sie einem Nutzungsplan einen API-Schlüssel hinzugefügt haben, kann es einige Minuten dauern, bis der Aktualisierungsvorgang abgeschlossen ist.

Schritte zur Konfiguration eines Nutzungsplans

In den folgenden Schritten wird beschrieben, wie Sie als API-Eigentümer einen Nutzungsplan für Ihre Kunden erstellen und konfigurieren.

So konfigurieren Sie einen Nutzungsplan

1. Erstellen Sie eine oder mehrere APIs, konfigurieren Sie die Methoden so, dass ein API-Schlüssel erforderlich ist, und stellen Sie die APIs in Stufen bereit.
2. Generieren oder importieren Sie API-Schlüssel und verteilen Sie diese an die Anwendungsentwickler (Ihre Kunden), die Ihre API einsetzen werden.
3. Erstellen Sie einen Nutzungsplan mit den gewünschten Drosselungs- und Kontingentlimits.
4. Ordnen Sie dem Nutzungsplan API-Stufen und API-Schlüssel zu.

API-Aufrufer müssen bei Anforderungen an diese API einen zugewiesenen API-Schlüssel im Header `x-api-key` aufweisen.

> **Note**
>
> Um API-Methoden in einem Nutzungsplan aufzunehmen, müssen einzelne API-Methoden so konfiguriert werden, dass ein API-Schlüssel verlangt wird. Zu berücksichtigende bewährte Methoden finden Sie unter the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”.

Auswählen einer API-Schlüsselquelle

Wenn Sie einen Nutzungsplan einer API zuordnen und API-Schlüssel bei API-Methoden aktivieren, muss jede eingehende API-Anfrage einen API-Schlüssel enthalten. API Gateway liest den Schlüssel und vergleicht ihn mit den Schlüsseln im Nutzungsplan. Wenn die Schlüssel übereinstimmen, werden die Anforderungen entsprechend der Anforderungslimits und des Kontingents des Plans von API Gateway gedrosselt. Andernfalls wird eine `InvalidKeyParameter`-Ausnahme ausgelöst. Dadurch erhält der Aufrufer eine `403 Forbidden`-Antwort.

Ihre API Gateway-API kann API-Schlüssel aus einer von zwei Quellen erhalten:

`HEADER`

Sie verteilen API-Schlüssel an Ihre Kunden und erfordern die Übermittlung des API-Schlüssels als `X-API-Key`-Header bei jeder eingehenden Anforderung.

`AUTHORIZER`

Sie lassen einen Lambda-Genehmiger den API-Schlüssel im Rahmen der Autorisierungsantwort zurückgeben. Weitere Informationen zur Autorisierungsantwort finden Sie unter the section called “Ausgabe von einem API Gateway Lambda Authorizer”.

> **Note**
>
> Zu berücksichtigende bewährte Methoden finden Sie unter the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”.

So wählen Sie eine API-Schlüsselquelle für eine API über die API-Gateway-Konsole aus:

1. Melden Sie sich bei der API Gateway-Konsole an.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.
3. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
4. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
5. Wählen Sie unter API-Schlüsselquelle die Option `Header` oder `Authorizer` aus der Drop-down-Liste aus.
6. Wählen Sie Änderungen speichern aus.

Um mithilfe von eine API-Schlüsselquelle für eine API auszuwählen AWS CLI, rufen Sie den `update-rest-api`Befehl wie folgt auf:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
 op=replace,path=/apiKeySource,value=AUTHORIZER
```

Damit der Client einen API-Schlüssel sendet, legen Sie den `value` im vorherigen CLI-Befehl auf `HEADER` fest.

Wenn Sie eine API-Schlüsselquelle für eine API über die API Gateway-REST-API auswählen möchten, rufen Sie wie folgt `restapi:update` auf:

```
PATCH /restapis/fugvjdxtri/ HTTP/1.1
Content-Type: application/json
Host: apigateway.us-east-1.amazonaws.com
X-Amz-Date: 20160603T205348Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160603/us-east-1/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature={sig4_hash}

{
  "patchOperations" : [
    {
        "op" : "replace",
        "path" : "/apiKeySource",
        "value" : "HEADER"
    }
  ]
}
```

Damit ein Genehmiger einen API-Schlüssel zurückgibt, legen Sie den `value` in der vorherigen `AUTHORIZER`-Eingabe auf `patchOperations` fest.

Abhängig von dem von Ihnen gewählten API-Schlüsselquelltyp verwenden Sie eines der folgenden Verfahren für die Verwendung eines Header-bezogenen API-Schlüssels oder eines vom Genehmiger zurückgegebenen API-Schlüssels für einen Methodenaufruf:

Um Header-bezogene API-Schlüssel zu verwenden:

1. Erstellen Sie eine API mit den gewünschten API-Methoden und stellen Sie die API dann für eine Stufe bereit.
2. Erstellen Sie einen neuen Nutzungsplan oder wählen Sie einen bereits vorhandenen Plan aus. Fügen Sie die bereitgestellte API-Stufe dem Nutzungsplan hinzu. Ordnen Sie dem Nutzungsplan einen API-Schlüssel hinzu oder wählen Sie einen im Plan vorhandenen API-Schlüssel. Notieren Sie den ausgewählten API-Schlüsselwert.
3. Richten Sie API-Methoden ein, um einen API-Schlüssel zu fordern.
4. Stellen Sie die API derselben Stufe bereit. Wenn Sie die API einer neuen Stufe bereitstellen, stellen Sie sicher, dass Sie den Nutzungsplan aktualisieren, um ihn der neuen API-Stufe zuzuordnen.

Der Client kann nun die API-Methoden aufrufen, während er den `x-api-key`-Header mit dem gewählten API-Schlüssel als Header-Wert bereitstellt.

Um Genehmiger-bezogene API-Schlüssel zu verwenden:

1. Erstellen Sie eine API mit den gewünschten API-Methoden und stellen Sie die API dann für eine Stufe bereit.
2. Erstellen Sie einen neuen Nutzungsplan oder wählen Sie einen bereits vorhandenen Plan aus. Fügen Sie die bereitgestellte API-Stufe dem Nutzungsplan hinzu. Ordnen Sie dem Nutzungsplan einen API-Schlüssel hinzu oder wählen Sie einen im Plan vorhandenen API-Schlüssel. Notieren Sie den ausgewählten API-Schlüsselwert.
3. Erstellen eines token-basierten Lambda-Genehmigers. `usageIdentifierKey:{api-key}` als Root-Level-Eigenschaft der Autorisierungsantwort einschließen. Anweisungen zum Erstellen eines Token-basierten Autorisierers finden Sie unter. the section called “Beispiel für eine TOKEN Autorisierungs-Lambda-Funktion”
4. Richten Sie API-Methoden ein, um einen API-Schlüssel zu fordern, und aktivieren Sie den Lambda-Genehmiger für die Methoden.
5. Stellen Sie die API derselben Stufe bereit. Wenn Sie die API einer neuen Stufe bereitstellen, stellen Sie sicher, dass Sie den Nutzungsplan aktualisieren, um ihn der neuen API-Stufe zuzuordnen.

Der Client kann nun die Methoden aufrufen, für die ein API-Schlüssel gefordert wird, ohne explizit einen API-Schlüssel bereitstellen zu müssen. Der vom Genehmiger zurückgegebene API-Schlüssel wird automatisch verwendet.

Einrichten von API-Schlüsseln mit der API-Gateway-Konsole

Um API-Schlüssel einzurichten, gehen Sie wie folgt vor:

- Konfigurieren Sie API-Methoden, um einen API-Schlüssel erforderlich zu machen.
- Erstellen oder importieren Sie einen API-Schlüssel für die API in einer Region.

Vor der Einrichtung von API-Schlüsseln müssen Sie eine API erstellt haben und diese in einer Stufe bereitgestellt haben. Nach der Erstellung eines API-Schlüsselwerts kann dieser nicht geändert werden.

Anweisungen zum Erstellen und Bereitstellen einer API mithilfe der API Gateway-Konsole finden Sie unter REST-API in API Gateway entwickeln und Bereitstellen einer REST-API in Amazon API Gateway.

Nachdem Sie einen API-Schlüssel erstellt haben, müssen Sie ihn einem Nutzungsplan zuordnen. Weitere Informationen finden Sie unter Erstellen, Konfigurieren und Testen von Nutzungsplänen mit der API Gateway-Konsole.

> **Note**
>
> Zu berücksichtigende bewährte Methoden finden Sie unter the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”.

Themen

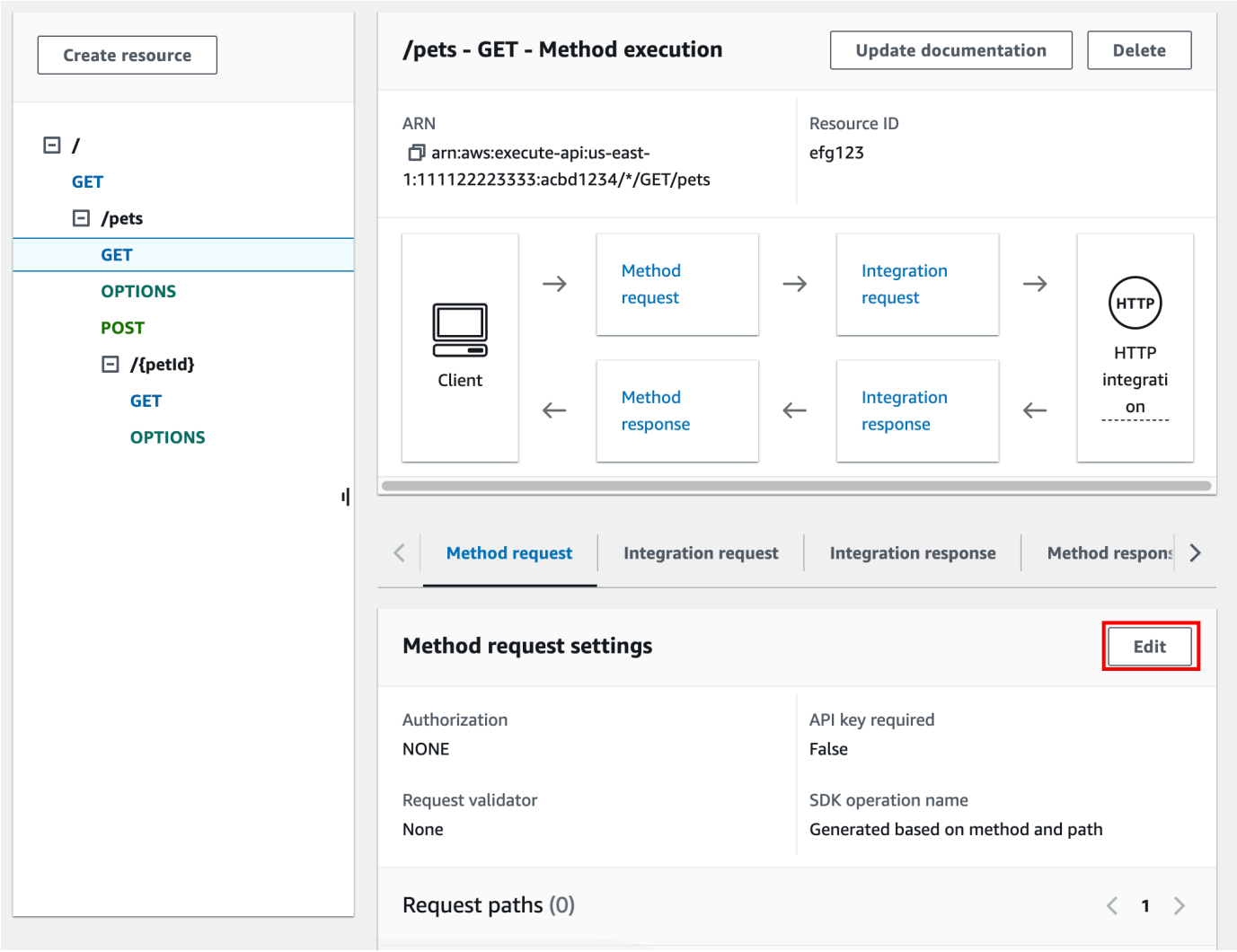
- Festlegen der Erforderlichkeit eines API-Schlüssels für eine Methode
- Erstellen eines API-Schlüssels
- Importieren von API-Schlüsseln

Festlegen der Erforderlichkeit eines API-Schlüssels für eine Methode

Im folgenden Verfahren wird beschrieben, wie Sie festlegen, dass für eine API-Methode ein API-Schlüssel erforderlich ist.

So konfigurieren Sie, dass für eine API-Methode ein API-Schlüssel erforderlich ist

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Resources (Ressourcen).
4. Erstellen Sie unter Resources eine neue Methode oder wählen Sie eine vorhandene aus.
5. Wählen Sie auf der Registerkarte Methodenanfrage unter Methodenanfrage-Einstellungen die Option Bearbeiten aus.



6. Wählen Sie API-Schlüssel erforderlich aus.
7. Wählen Sie Speichern.
8. Stellen Sie die API bereit (oder erneut bereit), damit die Änderung wirksam wird.

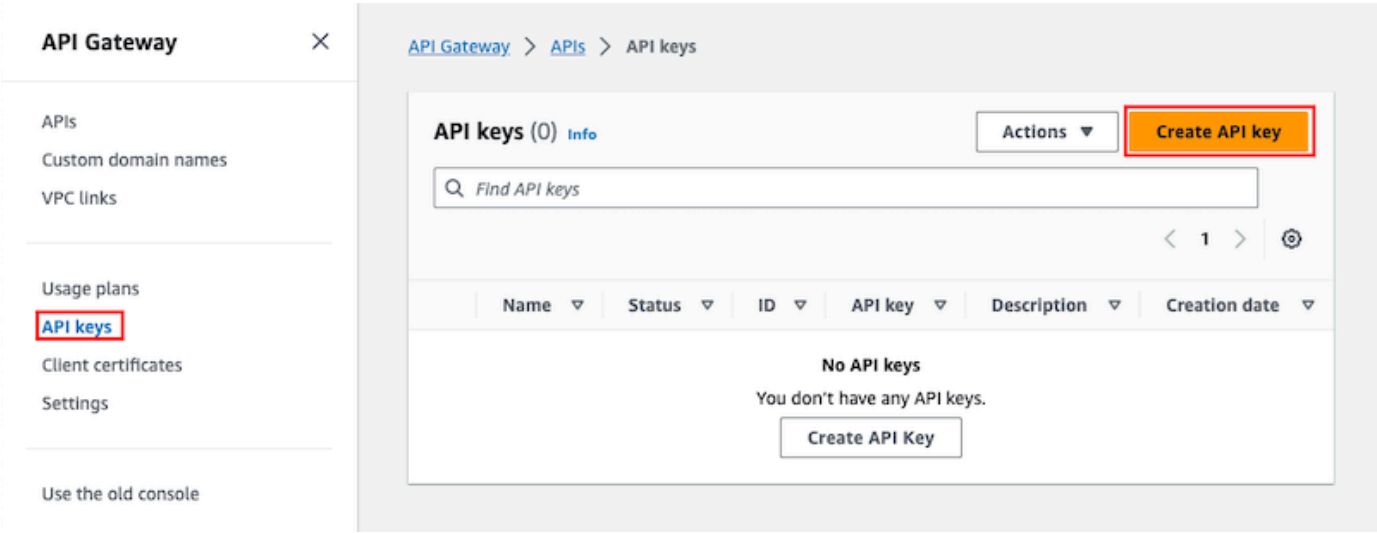
Wenn für die Option API-Schlüssel erforderlich der Wert `false` festgelegt ist und Sie die vorherigen Schritte nicht ausführen, wird keiner der API-Schlüssel, die einer API-Stufe zugeordnet sind, für diese Methode verwendet.

Erstellen eines API-Schlüssels

Sofern Sie bereits API-Schlüssel für die Nutzungspläne erstellt oder importiert haben, können Sie diesen und den nächsten Abschnitt überspringen.

So erstellen Sie einen API-Schlüssel

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich von API Gateway API-Schlüssel.
4. Klicken Sie auf API-Schlüssel erstellen.



5. Geben Sie unter Name einen Namen ein.
6. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
7. Wählen Sie für API-Schlüssel die Option Automatisch generieren, damit API Gateway den Schlüsselwert generiert, oder wählen Sie Benutzerdefiniert, um Ihren eigenen Schlüsselwert zu erstellen.
8. Wählen Sie Speichern.

Importieren von API-Schlüsseln

Im folgenden Verfahren wird beschrieben, wie Sie API-Schlüssel für Nutzungspläne importieren.

So importieren Sie API-Schlüssel

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich API-Schlüssel.

4. Wählen das im Dropdown-Menü Aktionen und dann die Option API-Schlüssel importieren.
5. Wählen Sie zum Laden einer CSV-Datei Datei auswählen aus. Sie können die Schlüssel auch im Texteditor eingeben. Weitere Informationen zum Dateiformat finden Sie unter the section called “API-Schlüsseldateiformat in API Gateway”.
6. Wählen Sie Fehler bei Warnungen aus, damit der Import im Falle eines Fehlers angehalten wird, oder wählen Sie Warnungen ignorieren aus, damit gültige Schlüssel auch bei einem auftretenden Fehler importiert werden.
7. Wählen Sie Import, um Ihre API-Schlüssel zu importieren.

Erstellen, Konfigurieren und Testen von Nutzungsplänen mit der API Gateway-Konsole

Vor der Erstellung eines Nutzungsplans sollten Sie sich vergewissern, dass die gewünschten API-Schlüssel konfiguriert sind. Weitere Informationen finden Sie unter Einrichten von API-Schlüsseln mit der API-Gateway-Konsole.

In diesem Abschnitt wird das Erstellen und Verwenden eines Nutzungsplans mithilfe der API Gateway-Konsole beschrieben.

Themen

- Migrieren Ihrer API auf Standard-Nutzungspläne (falls erforderlich)
- Erstellen eines Nutzungsplans
- Testen eines Nutzungsplans
- Verwalten eines Nutzungsplans

Migrieren Ihrer API auf Standard-Nutzungspläne (falls erforderlich)

Wenn Sie API Gateway nach Einführung der Nutzungsplanfunktionalität (11. August 2016) verwenden, sind für Sie automatisch Nutzungspläne in allen unterstützten Regionen aktiviert.

Wenn Sie API Gateway vor diesem Datum eingeführt haben, müssen Sie möglicherweise auf Standardnutzungspläne migrieren. Die Option Nutzungspläne aktivieren wird angezeigt, bevor Sie zum ersten Mal Nutzungspläne in der ausgewählten Region verwenden. Durch die Aktivierung dieser Option werden Standardnutzungspläne für jede einzelne API-Stufe erstellt, der vorhandene API-Schlüssel zugeordnet sind. Im Standard-Nutzungsplan sind anfänglich keine Drossel- oder Kontingentlimits festgelegt, und die Zuordnungen zwischen den API-Schlüsseln und API-Stufen werden in die Nutzungspläne kopiert. Das API-Verhalten bleibt unverändert. Sie müssen jedoch

die UsagePlanapiStagesEigenschaft verwenden, um die angegebenen API-Stufenwerte (apiIdundstage) den enthaltenen API-Schlüsseln (via UsagePlanKey) zuzuordnen, anstatt die ApiKeystageKeysEigenschaft zu verwenden.

Um zu überprüfen, ob Sie bereits auf Standardnutzungspläne migriert haben, führen Sie den CLI-Befehl get-account aus. In der Befehlsausgabe enthält die features-Liste den Eintrag "UsagePlans", wenn Nutzungspläne aktiviert sind.

Sie können Ihre APIs auch AWS CLI wie folgt auf Standard-Nutzungspläne migrieren:

Um zu Standard-Nutzungsplänen zu migrieren, verwenden Sie AWS CLI

1. Rufen Sie diesen CLI-Befehl auf: update-account.
2. Verwenden Sie für den cli-input-json-Parameter das folgende JSON-Konstrukt:

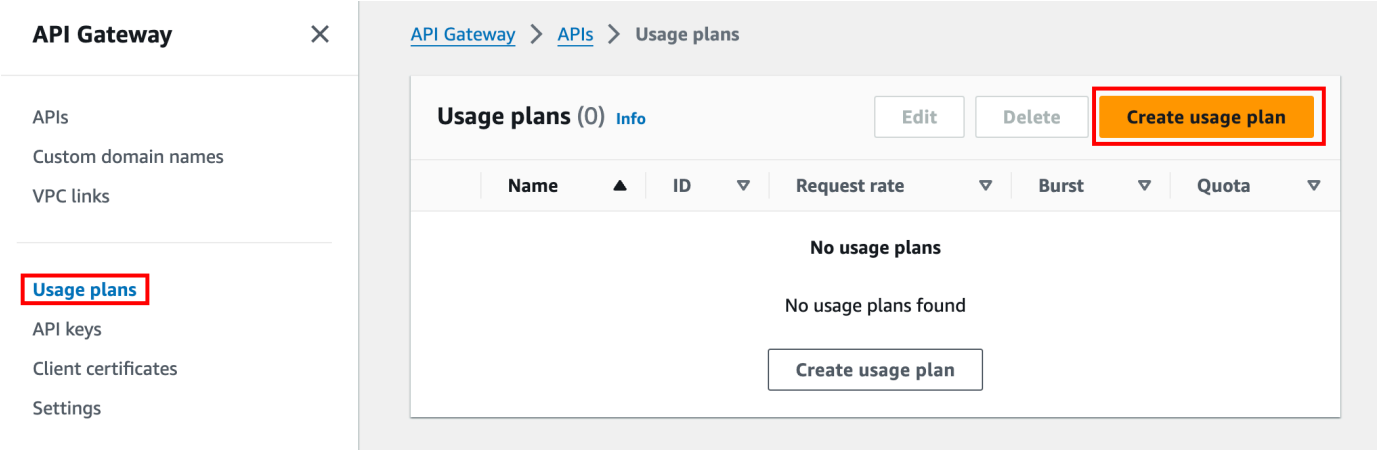
```
[
    {
        "op": "add",
        "path": "/features",
        "value": "UsagePlans"
    }
]
```

Erstellen eines Nutzungsplans

Im folgenden Verfahren wird das Erstellen eines Nutzungsplans beschrieben.

So erstellen Sie einen Nutzungsplan

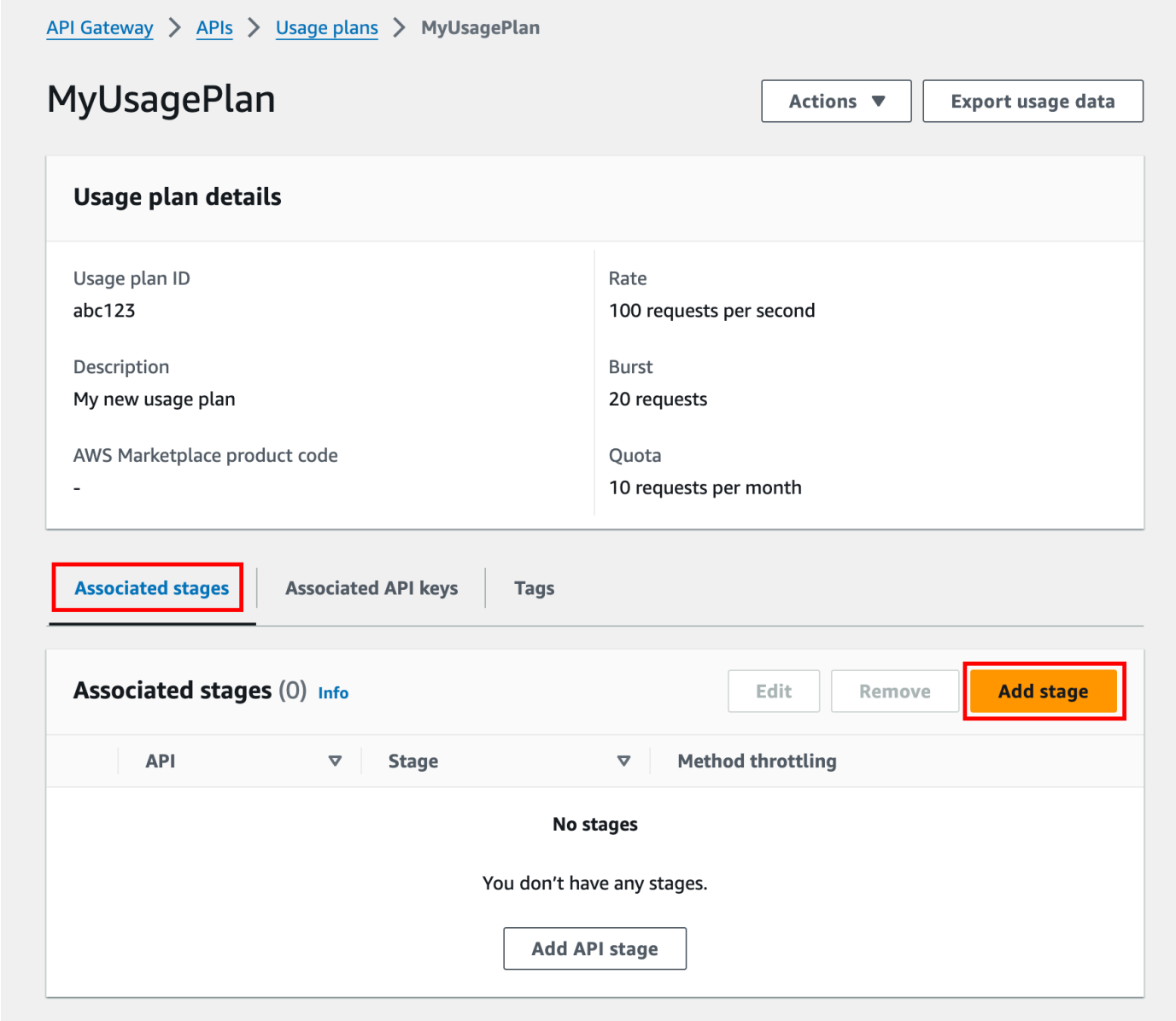
1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie im Hauptnavigationsbereich von API Gateway Nutzungspläne und dann Nutzungspläne erstellen.



3. Geben Sie unter Name einen Namen ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Standardmäßig ermöglichen Nutzungspläne die Drosselung. Geben Sie eine Rate und einen Burst-Wert für Ihren Nutzungsplan ein. Wählen Sie Drosselung, um die Drosselung auszuschalten.
6. Standardmäßig ermöglichen Nutzungspläne ein Kontingent für einen bestimmten Zeitraum. Geben Sie unter Anfragen die Gesamtzahl der Anfragen ein, die ein Benutzer im Zeitraum Ihres Nutzungsplans stellen kann. Wählen Sie Kontingent, um das Kontingent zu deaktivieren.
7. Wählen Sie Nutzungsplan erstellen.

So fügen Sie dem Nutzungsplan eine Stufe hinzu

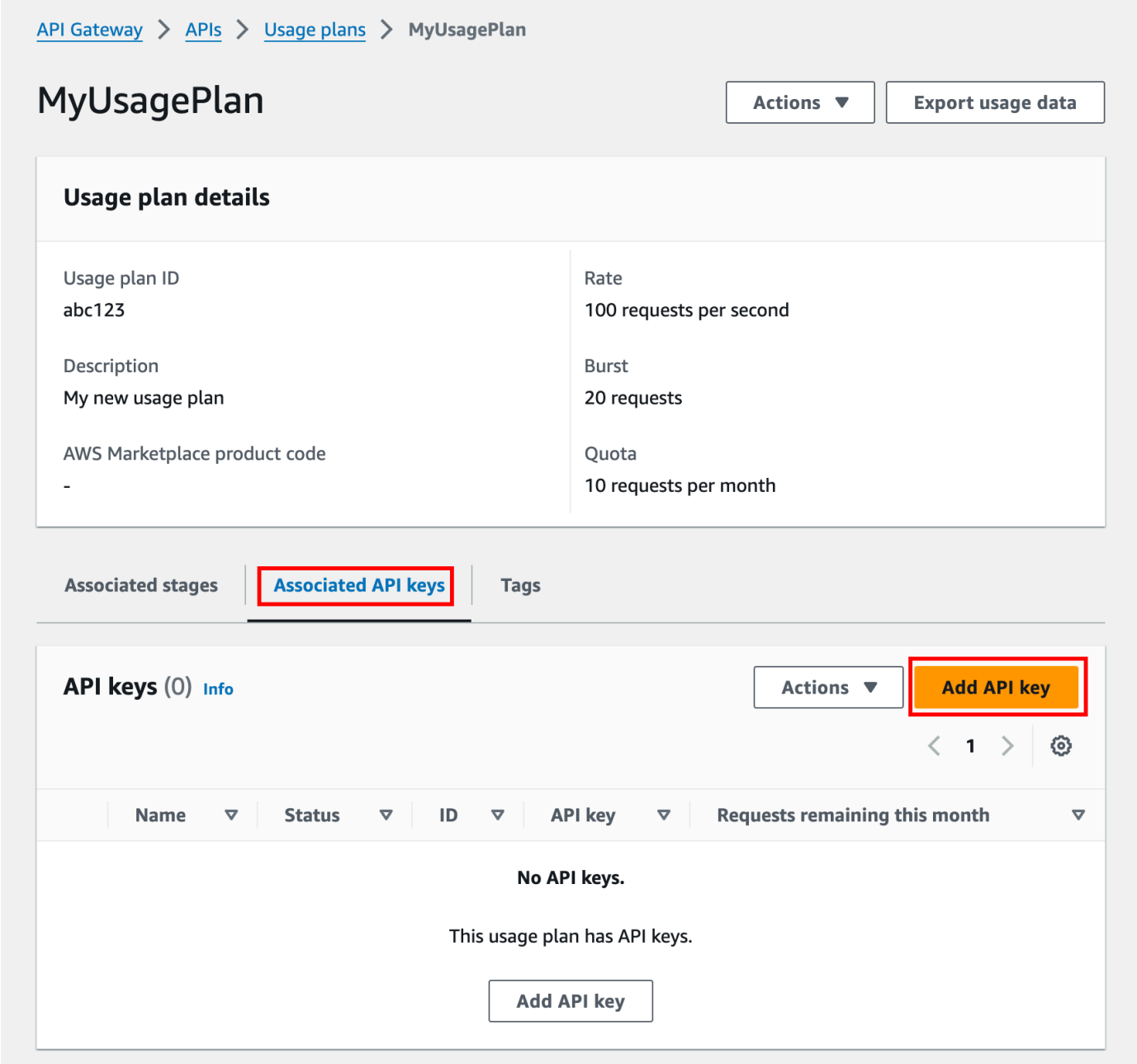
1. Wählen Sie Ihren Nutzungsplan aus.
2. Wählen Sie auf der Registerkarte Zugeordnete Stufen die Option Stufe hinzufügen aus.



3. Wählen Sie für API eine API aus.
4. Wählen Sie für Stufe eine Stufe aus.
5. (Optional) Gehen Sie wie folgt vor, um die Drosselung auf Methodenebene zu aktivieren:
   a. Wählen Sie Drosselung auf Methodenebene und anschließend Methode hinzufügen aus.
   b. Wählen Sie unter Ressource eine Ressource aus Ihrer API aus.
   c. Wählen Sie unter Methode eine Methode aus Ihrer API aus.
   d. Geben Sie eine Rate und einen Burst-Wert für Ihren Nutzungsplan ein.
6. Wählen Sie Zu Nutzungsplan hinzufügen aus.

So fügen dem Nutzungsplan einen Schlüssel hinzu

1. Wählen Sie auf der Registerkarte Zugeordnete API-Schlüssel die Option API-Schlüssel hinzufügen aus.



2. a. Um Ihrem Nutzungsplan einen vorhandenen Schlüssel zuzuordnen, wählen Sie Bestehenden Schlüssel hinzufügen und wählen Sie dann Ihren vorhandenen Schlüssel aus dem Drop-down-Menü aus.

b. Um einen neuen API-Schlüssel zu erstellen, wählen Sie Neuen Schlüssel erstellen und hinzufügen aus und erstellen Sie dann einen neuen Schlüssel. Weitere Informationen zum Erstellen eines neuen Schlüssels finden Sie unter Erstellen eines API-Schlüssels.

3. Wählen Sie API-Schlüssel hinzufügen.

Testen eines Nutzungsplans

Um den Nutzungsplan zu testen, können Sie ein AWS SDK oder einen REST-API-Client wie Postman verwenden. AWS CLI Ein Beispiel für die Verwendung von Postman zum Testen des Nutzungsplans finden Sie unter Testen von Nutzungsplänen.

Verwalten eines Nutzungsplans

Im Rahmen der Nutzungsplanverwaltung werden verwendete und verbleibende Kontingente in einem bestimmten Zeitraum überwacht und die verbleibenden Kontingente gegebenenfalls um eine angegebene Menge erhöht. In den folgenden Verfahren wird beschrieben, wie Sie Kontingente überwachen.

So überwachen Sie verwendete und verbleibende Kontingente

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Nutzungspläne.
3. Wählen Sie einen Nutzungsplan aus.
4. Wählen Sie die Registerkarte Zugeordnete API-Schlüssel, um die Anzahl der für den Zeitraum verbleibenden Anfragen für jeden Schlüssel anzuzeigen.
5. (Optional) Wählen Sie Nutzungsdaten exportieren und dann ein Startdatum und ein Enddatum aus. Wählen Sie dann JSON oder CSV als Exportdatenformat und anschließend Exportieren aus.

   Im folgenden Beispiel wird eine exportierte Datei angezeigt.

```
{
  "thisPeriod": {
    "px1KW6...qBazOJH": [
      [
        0,
        5000
```

```
      ],
      [
        0,
        5000
      ],
      [
        0,
        10
      ]
    ]
  },
  "startDate": "2016-08-01",
  "endDate": "2016-08-03"
}
```

Die Nutzungsdaten im Beispiel geben die tägliche Nutzung eines API-Clients, der vom API-Schlüssel (`px1KW6...qBaz0JH`) identifiziert wird, zwischen dem 1. August 2016 und dem 3. August 2016 an. In den täglichen Nutzungsdaten werden jeweils die verwendeten und die verbleibenden Kontingente angezeigt. In diesem Beispiel hat der Abonnent die ihm zugeordneten Kontingente gar nicht verwendet und der API-Besitzer oder Administrator hat das verbleibende Kontingent für den dritten Tag von 5.000 auf 10 gesenkt.

In den folgenden Verfahren wird beschrieben, wie Sie Kontingente ändern.

So erhöhen Sie verbleibende Kontingente

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Klicken Sie im Hauptnavigationsbereich von API Gateway auf Nutzungspläne.
3. Wählen Sie einen Nutzungsplan aus.
4. Wählen Sie die Registerkarte Zugeordnete API-Schlüssel, um die Anzahl der für den Zeitraum verbleibenden Anfragen für jeden Schlüssel anzuzeigen.
5. Wählen Sie einen API-Schlüssel und dann Nutzungserweiterung gewähren aus.
6. Geben Sie eine Zahl für das Kontingent Verbleibende Anforderungen an. Sie können die Anzahl der Umbenennungsanfragen erhöhen oder die Anzahl der verbleibenden Anfragen für den Zeitraum Ihres Nutzungsplans verringern.
7. Wählen Sie Kontingent aktualisieren aus.

Einrichten von API-Schlüsseln mit der API Gateway-REST-API

Um API-Schlüssel einzurichten, gehen Sie wie folgt vor:

- Konfigurieren Sie API-Methoden, um einen API-Schlüssel erforderlich zu machen.
- Erstellen oder importieren Sie einen API-Schlüssel für die API in einer Region.

Vor der Einrichtung von API-Schlüsseln müssen Sie eine API erstellt haben und diese in einer Stufe bereitgestellt haben. Nach der Erstellung eines API-Schlüsselwerts kann dieser nicht geändert werden.

Informationen über REST-API-Aufrufe zum Erstellen und Bereitstellen einer API finden Sie unter `restapi:create` und `deployment:create`.

> **Note**
>
> Zu berücksichtigende bewährte Methoden finden Sie unter the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”.

Themen

Festlegen der Erforderlichkeit eines API-Schlüssels für eine Methode

Führen Sie einen der folgenden Schritte aus, damit für eine Methode ein API-Schlüssel erforderlich ist:

- Rufen Sie `method:put` auf, um eine Methode zu erstellen. Legen Sie `apiKeyRequired` in der Anforderungsnutzlast auf `true` fest.
- Rufen Sie `method:update` auf, um `apiKeyRequired` auf `true` zu setzen.

Erstellen oder Importieren von API-Schlüsseln

Führen Sie einen der folgenden Schritte aus, um einen API-Schlüssel zu erstellen oder zu importieren:

- Rufen Sie `apikey:create` auf, um einen API-Schlüssel zu erstellen.
- Rufen Sie `apikey:import` auf, um einen API-Schlüssel aus einer Datei zu importieren. Weitere Informationen zum Dateiformat finden Sie unter API-Schlüsseldateiformat in API Gateway.

Sie können den Wert des neuen API-Schlüssels nicht ändern. Weitere Informationen zum Konfigurieren eines Nutzungsplans finden Sie unter Erstellen, Konfigurieren und Testen von Nutzungsplänen mittels API Gateway-CLI und REST-API.

Erstellen, Konfigurieren und Testen von Nutzungsplänen mittels API Gateway-CLI und REST-API

Sie können den Nutzungsplan erst konfigurieren, nachdem Sie Folgendes erledigt haben: die Methoden einer ausgewählten API so eingerichtet haben, dass API-Schlüssel erforderlich sind, die API für eine Stufe bereitgestellt oder erneut bereitgestellt haben und einen oder mehrere API-Schlüssel erstellt oder importiert haben. Weitere Informationen finden Sie unter Einrichten von API-Schlüsseln mit der API Gateway-REST-API.

Führen Sie die folgenden Anweisungen aus, um einen Nutzungsplan mit der API Gateway-REST-API zu konfigurieren (es wird davon ausgegangen, dass Sie die APIs, die dem Nutzungsplan hinzugefügt werden sollen, bereits erstellt haben).

Themen

- Migrieren zu Standardnutzungsplänen
- Erstellen eines Nutzungsplans
- Einen Nutzungsplan mithilfe der AWS CLI verwalten
- Testen von Nutzungsplänen

Migrieren zu Standardnutzungsplänen

Bei der ersten Erstellung eines Nutzungsplans können vorhandene API-Stufen, denen bestimmte API-Schlüssel zugeordnet sind, durch den Aufruf von `account:update` mit folgendem Inhalt zum Nutzungsplan migriert werden:

```
{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/features",
```

```
    "value" : "UsagePlans"
  } ]
}
```

Weitere Informationen zum Migrieren von API-Stufen mit zugeordneten API-Schlüsseln finden Sie unter Migrieren zu Standardnutzungsplänen in der API Gateway-Konsole.

Erstellen eines Nutzungsplans

Im folgenden Verfahren wird das Erstellen eines Nutzungsplans beschrieben.

So erstellen Sie einen Nutzungsplan mit der REST-API

1. Rufen Sie `usageplan:create` auf, um einen Nutzungsplan zu erstellen. Geben Sie in der Nutzlast den Namen und die Beschreibung des Plans, die zugehörigen API-Stufen, Bandbreitenlimits und Kontingente an.

   Notieren Sie die ID des Nutzungsplans. Sie benötigen diese im nächsten Schritt.
2. Führen Sie eine der folgenden Aufgaben aus:

   a. Rufen Sie `usageplankey:create` auf, um dem Nutzungsplan einen API-Schlüssel hinzuzufügen. Geben Sie in der Nutzlast `keyId` und `keyType` an.

      Wenn dem Nutzungsplan weitere API-Schlüssel hinzugefügt werden sollen, wiederholen Sie den vorherigen Aufruf für jeden API-Schlüssel.

   b. Rufen Sie `apikey:import` auf und fügen Sie einen oder mehrere API-Schlüssel direkt zum angegebenen Nutzungsplan hinzu. Die Anforderungsnutzlast sollte API-Schlüsselwerte, die zugehörige Nutzungsplan-ID, boolesche Flags (um anzugeben, dass die Schlüssel für den Nutzungsplan aktiviert sind) und ggf. Namen und Beschreibungen der API-Schlüssel enthalten.

      Im folgenden Beispiel der `apikey:import`-Anforderung werden drei API-Schlüssel (identifiziert durch `key`, `name` und `description`) zu einem Nutzungsplan (identifiziert durch `usageplanIds`) hinzugefügt:

      ```
      POST /apikeys?mode=import&format=csv&failonwarnings=fase HTTP/1.1
      Host: apigateway.us-east-1.amazonaws.com
      Content-Type: text/csv
      Authorization: ...
      ```

```
key,name, description, enabled, usageplanIds
abcdef1234ghijklmnop8901234567, importedKey_1, firstone,  tRuE, n371pt
abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt
abcdef1234ghijklmnop9012345678, importedKey_3,          , true, n371pt
```

Als Ergebnis werden drei `UsagePlanKey`-Ressourcen erstellt und zu `UsagePlan` hinzugefügt.

Auf diese Weise können Sie auch API-Schlüssel zu mehreren Nutzungsplänen hinzufügen. Dazu ändern Sie jeden `usageplanIds`-Spaltenwert in eine CSV-Zeichenfolge, die die ausgewählten Nutzungsplan-IDs in Anführungszeichen angibt (`"n371pt,m282qs"` oder `'n371pt,m282qs'`).

> **Note**
>
> Ein API-Schlüssel kann mehreren Nutzungsplänen zugeordnet werden. Ein Nutzungsplan kann mehreren Stufen zugeordnet werden. Ein bestimmter API-Schlüssel kann jedoch nur einem Nutzungsplan für jede Stufe Ihres API zugeordnet werden.

Einen Nutzungsplan mithilfe der AWS CLI verwalten

Die folgenden Codebeispiele zeigen, wie Sie die Drosselungseinstellungen auf Methodenebene in einem Nutzungsplan hinzufügen, entfernen oder modifizieren können, indem Sie den Befehl `update-usage-plan` aufrufen.

> **Note**
>
> Achten Sie darauf, `us-east-1` auf den korrekten Regionswert für Ihre API zu ändern.

Um ein Ratenlimit zur Drosselung einer individuellen Ressource und Methode hinzuzufügen oder zu ersetzen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations
                        op="replace",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>/rateLimit",value="0.1"
```

Um ein Burst-Limit zur Drosselung einer individuellen Ressource und Methode hinzuzufügen oder zu ersetzen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId>
 --patch-operations op="replace",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>/burstLimit",value="1"
```

Um die Drosseleinstellungen auf Methodenebene für eine individuelle Ressource und Methode zu entfernen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId>
 --patch-operations op="remove",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>",value=""
```

Um alle Drosseleinstellungen einer API auf Methodenebene zu entfernen:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations op="remove",path="/apiStages/<apiId>:<stage>/throttle ",value=""
```

Hier sehen Sie ein Beispiel unter Verwendung der Pet Store-Beispiel-API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations
                op="replace",path="/apiStages/<apiId>:<stage>/throttle",value='"{\"/
pets/GET\":{\"rateLimit\":1.0,\"burstLimit\":1},\"//GET\":{\"rateLimit\":1.0,
\"burstLimit\":1}}"'
```

Testen von Nutzungsplänen

Verwenden wir als Beispiel die PetStore API, die in erstellt wurdeTutorial: Erstellen einer REST-API durch Importieren eines Beispiels. Die API ist für die Verwendung des API-Schlüssels konfigurier `Hiorr45VR...c4GJc`. In den folgenden Schritten wird das Testen eines Nutzungsplans beschrieben.

So testen Sie den Nutzungsplan

- Führen Sie eine `GET`-Anforderung für die Ressource "Pets" (`/pets`) mit den Abfrageparametern `?type=...&page=...` der API (zum Beispiel `xbvxlpijch`) in einem Nutzungsplan aus:

  ```
  GET /testStage/pets?type=dog&page=1 HTTP/1.1
  ```

```
x-api-key: Hiorr45VR...c4GJc
Content-Type: application/x-www-form-urlencoded
Host: xbvxlpijch.execute-api.ap-southeast-1.amazonaws.com
X-Amz-Date: 20160803T001845Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160803/ap-southeast-1/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date;x-api-key,
 Signature={sigv4_hash}
```

> **Note**
>
> Sie müssen diese Anforderung an die Komponente `execute-api` von API Gateway übermitteln und den erforderlichen API-Schlüssel (zum Beispiel `Hiorr45VR...c4GJc`) im entsprechenden `x-api-key`-Header bereitstellen.

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit den angeforderten Ergebnissen aus dem Backend zurückgegeben. Falls Sie den `x-api-key`-Header nicht oder mit einem falschen Schlüssel gesetzt haben, erhalten Sie als Antwort die Meldung `403 Forbidden`. Wenn Sie allerdings nicht festgelegt haben, dass für die Methode ein API-Schlüssel erforderlich ist, erhalten Sie in jedem Fall die Antwort `200 OK` – unabhängig davon, ob der `x-api-key`-Header korrekt oder fehlerhaft eingerichtet wurde – und die Ablehnungs- und Kontingentlimits werden ignoriert.

Im Falle, dass API Gateway aufgrund eines internen Fehlers die im Nutzungsplan enthaltenen Drosselungs- und Kontingentlimits für diese Anforderung nicht erzwingen kann, führt API Gateway die Anforderung ohne Berücksichtigung der Limits aus. Es protokolliert jedoch eine Fehlermeldung von `Usage Plan check failed due to an internal error` in CloudWatch. Diese gelegentlich auftretenden Fehler können Sie ignorieren.

Erstellen und konfigurieren Sie API-Schlüssel und Nutzungspläne mit AWS CloudFormation

Sie können AWS CloudFormation sie verwenden, um API-Schlüssel für API-Methoden vorzuschreiben und einen Nutzungsplan für eine API zu erstellen. Die AWS CloudFormation Beispielvorlage macht Folgendes:

- Mit den Methoden `GET` und `POST` wird eine API-Gateway-API erstellt.

- Erfordert einen API-Schlüssel für die Methoden `GET` und `POST`. Diese API ruft Schlüssel aus dem `X-API-KEY`-Header jeder eingehenden Anfrage ab.
- Erstellt einen API-Schlüssel
- Es wird ein Nutzungsplan erstellt, um ein monatliches Kontingent von 1 000 Anfragen pro Monat, ein Limit bei der Drosselungsrate von 100 Anfragen pro Sekunde und ein Drosselungs-Burst-Limit von 200 Anfragen pro Sekunde festzulegen.
- Gibt für die `GET`-Methode ein Limit fvon 50 Anfragen pro Sekunde ür die Drosselungsrate auf Methodenebene und ein Burst-Limit für die Drosselung auf Methodenebene von 100 Anfragen pro Sekunde an.
- Ordnet dem Nutzungsplan API-Stufen und API-Schlüssel zu

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
  KeyName:
    Type: String
    Default: MyKeyName
    Description: Name of an API key
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: keys-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
```

```
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
  UsagePlan:
    Type: AWS::ApiGateway::UsagePlan
    DependsOn:
      - ApiDeployment
    Properties:
      Description: Example usage plan with a monthly quota of 1000 calls and method-
level throttling for /pets GET
      ApiStages:
        - ApiId: !Ref Api
          Stage: !Sub '${StageName}'
          Throttle:
            "/pets/GET":
              RateLimit: 50.0
              BurstLimit: 100
      Quota:
        Limit: 1000
        Period: MONTH
      Throttle:
        RateLimit: 100.0
        BurstLimit: 200
```

```
      UsagePlanName: "My Usage Plan"
  ApiKey:
    Type: AWS::ApiGateway::ApiKey
    Properties:
      Description: API Key
      Name: !Sub '${KeyName}'
      Enabled: True
  UsagePlanKey:
    Type: AWS::ApiGateway::UsagePlanKey
    Properties:
      KeyId: !Ref ApiKey
      KeyType: API_KEY
      UsagePlanId: !Ref UsagePlan
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Konfigurieren Sie eine Methode zur Verwendung von API-Schlüsseln mit einer OpenAPI-Definition

Sie können eine OpenAPI-Definition verwenden, um API-Schlüssel für eine Methode anzufordern.

Erstellen Sie für jede Methode ein Sicherheitsanforderungsobjekt, das einen API-Schlüssel zum Aufrufen dieser Methode erfordert. Definieren Sie es dann `api_key` in der Sicherheitsdefinition. Nachdem Sie Ihre API erstellt haben, fügen Sie die neue API-Phase zu Ihrem Nutzungsplan hinzu.

Das folgende Beispiel erstellt eine API und erfordert einen API-Schlüssel für die `GET` Methoden `POST` und:

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "2024-03-14T20:20:12Z",
    "title" : "keys-api"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/pets" : {
```

```
      "get" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      },
      "post" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "securityDefinitions" : {
    "api_key" : {
      "type" : "apiKey",
      "name" : "x-api-key",
      "in" : "header"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "keys-api",
    "version" : "2024-03-14T20:20:12Z"
  },
```

```
"servers" : [ {
  "url" : "{basePath}",
  "variables" : {
    "basePath" : {
      "default" : "v1"
    }
  }
} ],
"paths" : {
  "/pets" : {
    "get" : {
      "security" : [ {
        "api_key" : [ ]
      } ],
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
        "passthroughBehavior" : "when_no_match",
        "type" : "http_proxy"
      }
    },
    "post" : {
      "security" : [ {
        "api_key" : [ ]
      } ],
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
        "passthroughBehavior" : "when_no_match",
        "type" : "http_proxy"
      }
    }
  }
},
"components" : {
  "securitySchemes" : {
    "api_key" : {
      "type" : "apiKey",
      "name" : "x-api-key",
      "in" : "header"
    }
  }
}
```

```
}
```

API-Schlüsseldateiformat in API Gateway

API Gateway kann API-Schlüssel aus externen Dateien im CSV-Format importieren und die Schlüssel dann einem oder mehreren Nutzungsplänen zuordnen. Die importierte Datei muss die Spalten `Name` und `Key` aufweisen. Bei Spaltenüberschriften wird z. B. die Groß-/Kleinschreibung nicht beachtet und die Spaltenreihenfolge ist beliebig:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

Ein `Key` Wert muss zwischen 20 und 128 Zeichen lang sein. Ein `Name`-Wert darf nicht mehr als 1024 Zeichen enthalten.

Eine API-Schlüsseldatei kann z. B. auch die Spalten `Description`, `Enabled`oder `UsagePlanIds` aufweisen:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Ist ein Schlüssel z. B. mehreren Nutzungsplänen zugeordnet, handelt es sich bei dem `UsagePlanIds`-Wert um eine CSV-Zeichenfolge, die die Nutzungsplan-IDs in (einfachen oder doppelten) Anführungszeichen angibt:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsr"
```

Nicht erkannte Spalten sind zulässig, werden aber ignoriert. Der Standardwert ist eine leere Zeichenfolge oder ein boolescher `true`-Wert.

Derselbe API-Schlüssel kann mehrmals importiert werden, dabei überschreibt die neueste Version die vorherige. Zwei API-Schlüssel sind identisch, wenn sie den gleichen `key`-Wert haben.

> **Note**
>
> Zu berücksichtigende bewährte Methoden finden Sie unter the section called “Bewährte Methoden für API-Schlüssel und Nutzungspläne”.

Dokumentieren von REST-APIs

Um Kunden dabei zu helfen, Ihre API zu verstehen und zu verwenden, sollten Sie die API dokumentieren. Dazu können Sie mithilfe von API Gateway Hilfeinhalte für einzelne API-Entitäten als festen Bestandteil Ihres API-Entwicklungsprozesses hinzufügen und aktualisieren. API Gateway speichert die Quellinhalte und ermöglicht es Ihnen, verschiedene Versionen der Dokumentation zu archivieren. Sie können eine Dokumentationsversion mit einer API-Phase verknüpfen, einen Dokumentations-Snapshot zu einer bestimmten Phase in eine externe OpenAPI-Datei exportieren und die Datei zur Veröffentlichung der Dokumentation verteilen.

Um Ihre API zu dokumentieren, können Sie die API Gateway REST API aufrufen, eines der AWS SDKs verwenden, das AWS CLIfor API Gateway verwenden oder die API Gateway Gateway-Konsole verwenden. Sie haben auch die Möglichkeit, die in einer externen OpenAPI-Datei definierten Dokumentationsbausteine zu importieren oder zu exportieren.

Wenn Sie API-Dokumentation mit Entwicklern teilen möchten, können Sie ein Entwicklerportal verwenden. Ein Beispiel finden Sie im Blog AWS Partner Network (APN) unter Integration ReadMe mit API Gateway, um Ihren Developer Hub auf dem neuesten Stand zu halten.

Themen

- Darstellung der API-Dokumentation in API Gateway
- Dokumentieren einer API mit der API Gateway-Konsole
- Veröffentlichen der API-Dokumentation mithilfe der API Gateway-Konsole
- Dokumentieren einer API mit der API Gateway-REST-API
- Veröffentlichen der API-Dokumentation mithilfe der API Gateway-REST-API
- Importieren einer API-Dokumentation
- Steuern des Zugriffs auf Ihre API-Dokumentation

Darstellung der API-Dokumentation in API Gateway

Eine API Gateway-API-Dokumentation besteht aus einzelnen Bestandteilen (Bausteinen), die sich auf bestimmte API-Entitäten beziehen, z. B. API, Ressource, Methode, Anforderung, Antwort, Nachrichtenparameter (z. B. Pfad, Abfrage, Header) sowie Genehmiger und Modelle.

In API Gateway wird ein Dokumentationsteil durch eine DocumentationPartRessource dargestellt. Die API-Dokumentation als Ganzes wird durch die DocumentationPartsSammlung repräsentiert.

Die Dokumentation einer API umfasst das Erstellen von `DocumentationPart`-Instances, das Hinzufügen dieser Instances zur `DocumentationParts`-Sammlung und die Pflege der verschiedenen Versionen der Dokumentationsbausteine, die im API-Entwicklungsprozess entstehen.

Themen

- Dokumentationsbausteine
- Dokumentationsversionen

Dokumentationsbausteine

Eine DocumentationPartRessource ist ein JSON-Objekt, das den Dokumentationsinhalt speichert, der für eine einzelne API-Entität gilt. Das Feld `properties` enthält den Dokumentationsinhalt in Form einer Übersicht von Schlüssel-Wert-Paaren. Die Eigenschaft `location` identifiziert die zugehörige API-Entität.

Sie, der API-Entwickler, bestimmen die Form einer Content Map. Der Wert eines Schlüssel-Wert-Paars kann eine Zeichenfolge, eine Zahl, ein boolescher Wert, ein Objekt oder ein Array sein. Die Form des `location`-Objekts hängt vom jeweiligen Entitätstyp ab.

Die `DocumentationPart`-Ressource unterstützt das Vererben von Inhalten, das heißt, der Dokumentationsinhalt einer API-Entität gilt auch für die untergeordneten Elemente dieser API-Entität. Weitere Informationen zur Definition untergeordneter Entitäten und zur Vererbung von Inhalten finden Sie unter Inherit Content from an API Entität of More General Specification.

Lokation eines Dokumentationsbausteins

Die Location-Eigenschaft einer DocumentationPartInstanz identifiziert eine API-Entität, für die der zugehörige Inhalt gilt. Die API-Entität kann eine API-Gateway-REST-API-Ressource sein RestApi, z. B. Resource, Method MethodResponse, Authorizer oder Model. Die Entität kann auch ein Nachrichtenparameter sein, z. B. ein URL-Pfadparameter, ein Abfragezeichenfolge-Parameter, ein Anforderungs- oder Antwort-Header-Parameter, ein Anforderungs- oder Antworttext oder ein Antwortstatuscode.

Um eine API-Entität anzugeben, wählen Sie für das Attribut type des Objekts `location` eine der folgenden Optionen: `API`, `AUTHORIZER`, `MODEL`, `RESOURCE`, `METHOD`, `PATH_PARAMETER`, `QUERY_PARAMETER`, `REQUEST_HEADER`, `REQUEST_BODY`, `RESPONSE`, `RESPONSE_HEADER` oder `RESPONSE_BODY`.

Je nach `type` einer API-Entität können Sie auch andere `location`-Attribute angeben, z. B. method, name, path und statusCode. Nicht alle diese Attribute gelten für eine bestimmte API-Entität. Zum Beispiel sind `type`, `path`, `name` und `statusCode` gültige Attribute der Entität `RESPONSE`; nur `type` und `path` sind gültige Lokationsattribute der Entität `RESOURCE`. Wenn Sie ein ungültiges Feld in der `location` eines `DocumentationPart` für eine bestimmte API-Entität angeben, führt dies zu einem Fehler.

Nicht alle gültigen `location`-Felder sind erforderlich. Beispielsweise ist `type` ein gültiges und erforderliches `location`-Feld für alle API-Entitäten. Die Attribute `method`, `path` und `statusCode` hingegen sind gültige, aber keine erforderlichen Attribute für die Entität `RESPONSE`. Sofern nicht explizit angegeben, übernimmt ein gültiges `location`-Feld seinen Standardwert. Der Standardwert für `path` ist `/`, d. h. die Stammressource einer API. Der Standardwert für `method` oder `statusCode` ist `*`, d. h. jeder beliebige Methoden- bzw. Statuscodewert.

Inhalt eines Dokumentationsbausteins

Der Wert `properties` ist als JSON-Zeichenfolge kodiert. Der Wert `properties` enthält alle Informationen, die Sie auswählen, um Ihre Dokumentationsanforderungen zu erfüllen. Das folgende Beispiel zeigt eine gültige Content Map:

```
{
  "info": {
    "description": "My first API with Amazon API Gateway."
  },
  "x-custom-info" : "My custom info, recognized by OpenAPI.",
  "my-info" : "My custom info not recognized by OpenAPI."
}
```

Auch wenn API Gateway jede gültige JSON-Zeichenfolge als Content Map akzeptiert, werden die Inhaltsattribute als zwei Kategorien behandelt: als Attribute, die von OpenAPI erkannt werden, und als Attribute, die nicht von OpenAPI erkannt werden. Im vorherigen Beispiel werden `info`, `description` und `x-custom-info` von OpenAPI als OpenAPI-Standardobjekt, -eigenschaft oder -erweiterung erkannt. Im Gegensatz dazu ist `my-info` nicht mit der OpenAPI-Spezifikation kompatibel. API Gateway verteilt OpenAPI-kompatible Inhaltsattribute in die API-Entitätsdefinitionen von den verknüpften `DocumentationPart`-Instances. Die nicht kompatiblen Inhaltsattribute werden von API Gateway nicht in die API-Entitätsdefinitionen verteilt.

Ein weiteres Beispiel, bei dem `DocumentationPart` für eine `Resource`-Entität gilt:

```
{
```

```
    "location" : {
        "type" : "RESOURCE",
        "path": "/pets"
    },
    "properties" : {
        "summary" : "The /pets resource represents a collection of pets in PetStore.",
        "description": "... a child resource under the root...",
    }
}
```

Hier sind sowohl `type` als auch `path` gültige Felder zur Identifizierung des Ziels des Typs `RESOURCE`. Für die Stammressource (`/`) können Sie das Feld `path` weglassen.

```
{
    "location" : {
        "type" : "RESOURCE"
    },
    "properties" : {
        "description" : "The root resource with the default path specification."
    }
}
```

Dies entspricht der folgenden `DocumentationPart`-Instance:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/"
    },
    "properties" : {
        "description" : "The root resource with an explicit path specification"
    }
}
```

Vererben von Inhalten aus einer API-Entität allgemeinerer Spezifikationen

Der Standardwert eines optionalen `location`-Felds bietet eine musterhafte Beschreibung einer API-Entität. Unter Verwendung des Standardwerts im Objekt `location` können Sie eine allgemeine Beschreibung in der `properties`-Map zu einer `DocumentationPart`-Instance mit dieser Art von `location`-Muster hinzufügen. API Gateway extrahiert die entsprechenden OpenAPI-Dokumentationsattribute aus dem `DocumentationPart` der allgemeinen API-Entität und fügt sie in

eine spezifische API-Entität mit den `location`-Feldern ein, die dem allgemeinen `location`-Muster entsprechen oder mit dem genauen Wert übereinstimmen, es sei denn, der betreffenden Entität ist bereits eine `DocumentationPart`-Instance zugeordnet. Dieses Verhalten wird auch als Vererbung von Inhalten aus einer API-Entität allgemeinerer Spezifikationen bezeichnet.

Die Vererbung von Inhalten gilt für bestimmte API-Entitätentypen nicht. Weitere Details finden Sie in der nachfolgenden Tabelle.

Wenn eine API-Entität mit mehr als einem `DocumentationPart`-Lokationsmuster übereinstimmt, übernimmt die Entität den Dokumentationsbaustein mit den "location"-Feldern der höchsten Priorität und Spezifität. Die Rangfolge ist `path` > `statusCode`. Für den Abgleich mit dem `path`-Feld wählt API Gateway die Entität mit dem spezifischsten Pfadwert. In der folgenden Tabelle wird dies mit ein paar Beispielen veranschaulicht.

Fall	**path**	**statusCode**	**name**	Anmerken
1	`/pets`	`*`	`id`	Die mit diesem Lokationsmuster verknüpfte Dokumentation wird von den Entitäten übernommen, die mit dem Lokationsmuster

Fall	**path**	**statusCode**	**name**	Anmerkungen
				übereinstimmen.

Fall	**path**	**statusCode**	**name**	Anmerken
2	/pets	200	id	Die mit diesem Lokationsmuster verknüpfte Dokumentation wird von den Entitäten übernommen, die mit dem Lokationsmuster übereinstimmen, sofern Fall 1 und 2 abgeglichen werden, da Fall 2

Fall	**path**	**statusCode**	**name**	Anmerken
				spezifischer ist als Fall 1.

Fall	**path**	**statusCode**	**name**	Anmerken
3	/pets/petId	*	id	Die mit diesem Lokationsmuster verknüpfte Dokumentation wird von den Entitäten übernommen, die mit dem Lokationsmuster übereinstimmen, wenn Fall 1, 2 und 3 abgeglichen werden, da Fall 3

Fall	**path**	**statusCode**	**name**	Anmerkungen
				eine höhere Priorität hat als Fall 2 und spezifischer ist als Fall 1.

Hier ein weiteres Beispiel für eine allgemeinere `DocumentationPart`-Instance im Vergleich zu einer spezifischeren Instance. Die folgende allgemeine Fehlermeldung `"Invalid request error"` wird in die OpenAPI-Definitionen der `400`-Fehlermeldungen eingefügt, sofern sie nicht überschrieben werden.

```
{
    "location" : {
        "type" : "RESPONSE",
        "statusCode": "400"
    },
    "properties" : {
        "description" : "Invalid request error."
    }"
}
```

Mit der folgenden Überschreibung verfügen die `400`-Antworten auf alle Methoden der `/pets`-Ressource stattdessen über die Beschreibung `"Invalid petId specified"`.

```
{
    "location" : {
```

```
        "type" : "RESPONSE",
        "path": "/pets",
        "statusCode": "400"
    },
    "properties" : "{
        "description" : "Invalid petId specified."
    }"
}
```

Zulässige "location"-Felder für **`DocumentationPart`**

Die folgende Tabelle zeigt die gültigen und erforderlichen Felder sowie die geltenden Standardwerte einer DocumentationPartRessource, die einem bestimmten Typ von API-Entitäten zugeordnet ist.

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
API	`{ "location": { "type": "API" }, ... }`	`type`	–	Nein
Ressource	`{ "location": { "type": "RESOURCE", "path": "resource_path " }, ... }`	`type`	Der Standardwert von `path` ist `/`.	Nein
Methode	`{ "location": { "type": "METHOD",`	`type`	Die Standardwerte von `path` und `method` sind `/` bzw. `*`.	Ja, Abgleich von `path` nach Präfix und von `method`

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
	"path": "*resource_path* ", "method": "*http_verb* " }, ... }			nach beliebigen Werten
Abfrageparameter	{ "location": { "type": "QUERY_PARAMETER", "path": "*resource_path* ", "method": "*HTTP_verb* ", "name": "*query_parameter_name* " }, ... }	`type`	Die Standardwerte von `path` und `method` sind `/` bzw. `*`.	Ja, Abgleich von `path` nach Präfix und von `method` nach exakten Werten
Anforderungstext	{ "location": { "type": "REQUEST_BODY", "path": "*resource_path* ", "method": "*http_verb* " }, ... }	`type`	Die Standardwerte von `path` und `method` sind `/` bzw. `*`.	Ja, Abgleich von `path` nach Präfix und von `method` nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
Parameter des Anforderungs-Headers	`{ "location": { "type": "REQUEST_HEADER", "path": "resource_path ", "method": "HTTP_verb ", "name": "header_name " }, ... }`	`type`, `name`	Die Standardwerte von `path` und `method` sind `/` bzw. `*`.	Ja, Abgleich von `path` nach Präfix und von `method` nach exakten Werten
Pfadparameter der Anforderung	`{ "location": { "type": "PATH_PARAMETER", "path": "resource/{path_parameter_name }", "method": "HTTP_verb ", "name": "path_parameter_name " }, ... }`	`type`, `name`	Die Standardwerte von `path` und `method` sind `/` bzw. `*`.	Ja, Abgleich von `path` nach Präfix und von `method` nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
Antwort	`{ "location": { "type": "RESPONSE", "path": "resource_path", "method": "http_verb", "statusCode": "status_code" }, ... }`	`type`	Die Standardwerte von `path`, `method` und `statusCode` sind `/`, * bzw. *.	Ja, Abgleich von `path` nach Präfix und von `method` und `statusCode` nach exakten Werten
Antwort-Header	`{ "location": { "type": "RESPONSE_HEADER", "path": "resource_path", "method": "http_verb", "statusCode": "status_code", "name": "header_name" }, ... }`	`type`, `name`	Die Standardwerte von `path`, `method` und `statusCode` sind `/`, * bzw. *.	Ja, Abgleich von `path` nach Präfix und von `method` und `statusCode` nach exakten Werten

API-Entität	Gültige "location"-Felder	Erforderliche "location"-Felder	Standardfeldwerte	Vererbbare Inhalte
Antworttext	`{ "location": { "type": "RESPONSE_BODY", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	Die Standardwerte von `path`, `method` und `statusCode` sind `/`, * bzw. *.	Ja, Abgleich von `path` nach Präfix und von `method` und `statusCode` nach exakten Werten
Authorizer	`{ "location": { "type": "AUTHORIZER", "name": "authorizer_name " }, ... }`	`type`	–	Nein
Model	`{ "location": { "type": "MODEL", "name": "model_name " }, ... }`	`type`	–	Nein

Dokumentationsversionen

Eine Dokumentationsversion ist eine Momentaufnahme der DocumentationPartsSammlung einer API und ist mit einer Versionskennung gekennzeichnet. Die Veröffentlichung der Dokumentation einer API umfasst das Erstellen einer Dokumentationsversion, deren Verknüpfung mit einer API-Stufe und das Exportieren dieser stufenspezifischen Version der API-Dokumentation in eine externe OpenAPI-Datei. In API Gateway wird ein Dokumentations-Snapshot als DocumentationVersionRessource dargestellt.

Wenn Sie eine API aktualisieren, erstellen Sie neue Versionen der API. In API Gateway verwalten Sie alle Dokumentationsversionen mithilfe der DocumentationVersionsSammlung.

Dokumentieren einer API mit der API Gateway-Konsole

In diesem Abschnitt wird beschrieben, wie Sie die Dokumentationsbausteine einer API mit der API Gateway-Konsole erstellen und verwalten.

Eine Voraussetzung für das Erstellen und Bearbeiten der Dokumentation einer API ist, dass Sie die API bereits erstellt haben. In diesem Abschnitt verwenden wir die PetStoreAPI als Beispiel. Um eine API mit der API Gateway-Konsole zu erstellen, befolgen Sie die Anweisungen in Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Themen

- Dokumentieren der API-Entität
- Dokumentieren einer RESOURCE-Entität
- Dokumentieren einer METHOD-Entität
- Dokumentieren einer QUERY_PARAMETER-Entität
- Dokumentieren einer PATH_PARAMETER-Entität
- Dokumentieren einer REQUEST_HEADER-Entität
- Dokumentieren einer REQUEST_BODY-Entität
- Dokumentieren einer RESPONSE-Entität
- Dokumentieren einer RESPONSE_HEADER-Entität
- Dokumentieren einer RESPONSE_BODY-Entität
- Dokumentieren einer MODEL-Entität
- Dokumentieren einer AUTHORIZER-Entität

Dokumentieren der **API**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für die API-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option API aus.

   Wenn kein Dokumentationsbaustein für die API erstellt wurde, wird der properties-Map-Editor des Dokumentationsbausteins angezeigt. Geben Sie die folgende properties-Zuweisung in den Texteditor ein.

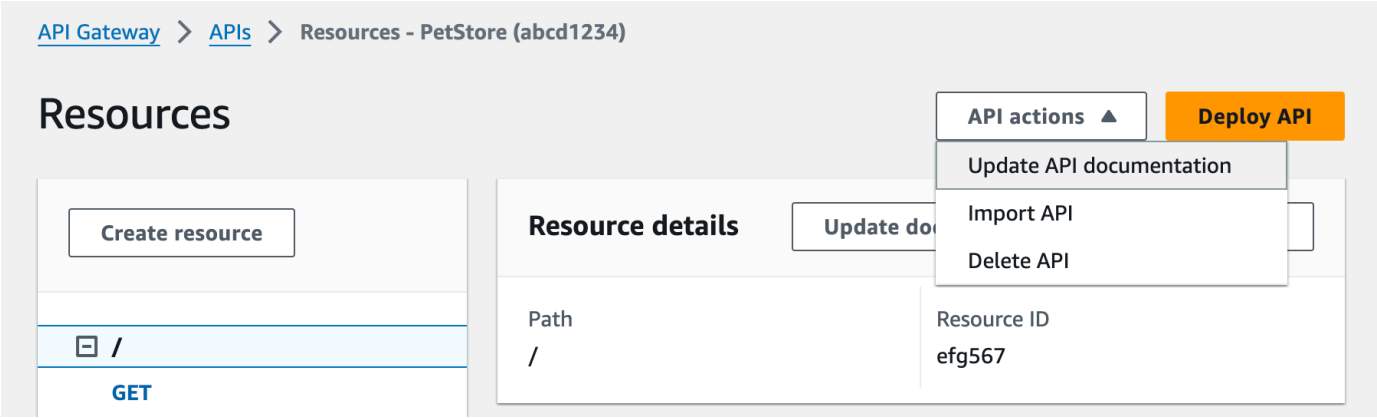
   ```
   {
     "info": {
       "description": "Your first API Gateway API.",
       "contact": {
           "name": "John Doe",
           "email": "john.doe@api.com"
       }
     }
   }
   ```

   > **Note**
   >
   > Sie müssen die properties-Zuordnung nicht als JSON-Zeichenfolge verschlüsseln. Die API Gateway-Konsole stringifiziert das JSON-Objekt für Sie.

3. Wählen Sie Create documentation part (Dokumentation erstellen) aus.

Gehen Sie wie folgt vor, um eine neue Dokumentation für die API-Entität im Bereich Resources (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
2. Wählen Sie das Menü API actions (API-Aktionen) und dann Update API documentation (API-Dokumentation aktualisieren) aus.



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Wählen Sie den Namen Ihrer API aus und klicken Sie dann auf der API-Karte auf Edit (Bearbeiten).

Dokumentieren einer **`RESOURCE`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `RESOURCE`-Entität hinzuzufügen:

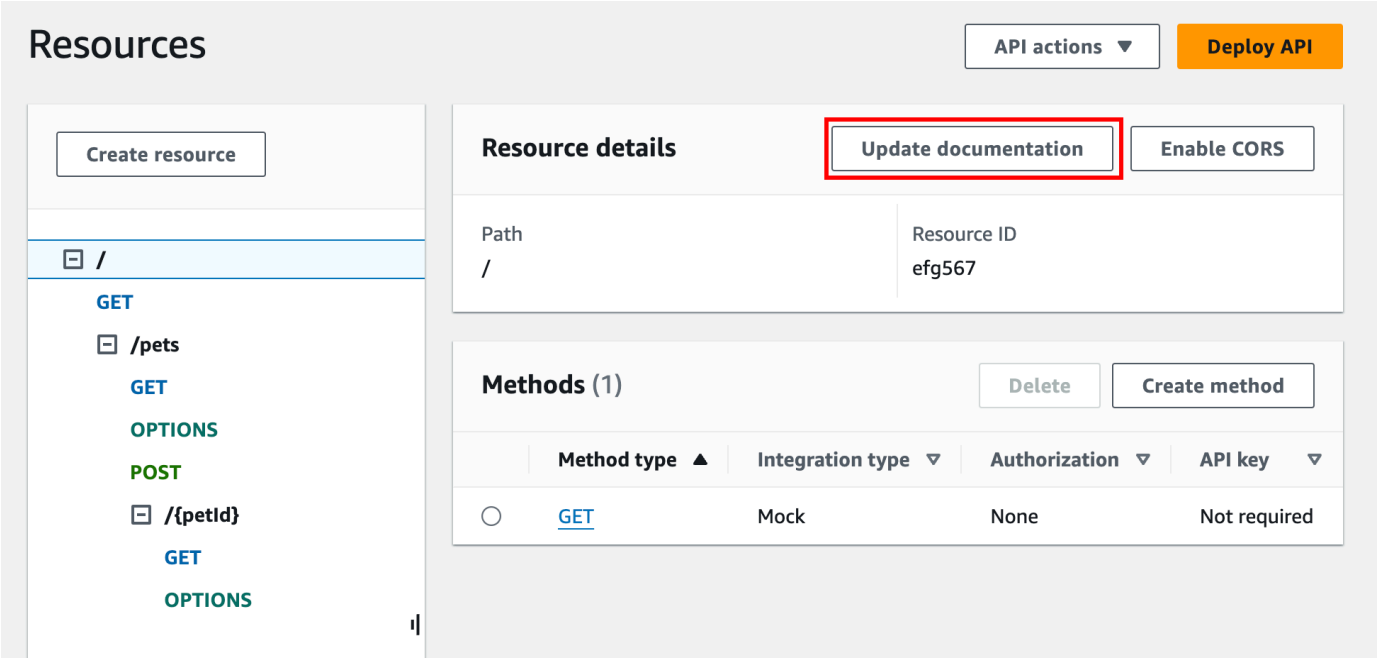
1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Resource (Ressource) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Geben Sie eine Beschreibung im Texteditor ein, zum Beispiel:

```
{
    "description": "The PetStore's root resource."
}
```

5. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für eine nicht aufgeführte Ressource erstellen.
6. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `RESOURCE`-Entität im Bereich Resources (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
2. Wählen Sie die Ressource aus und klicken Sie dann auf Update documentation (Dokumentation aktualisieren).



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Wählen Sie die Ressource aus, die Ihre Dokumentation enthält, und klicken Sie dann auf Edit (Bearbeiten).

Dokumentieren einer `METHOD`-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `METHOD`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Method (Methode) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.

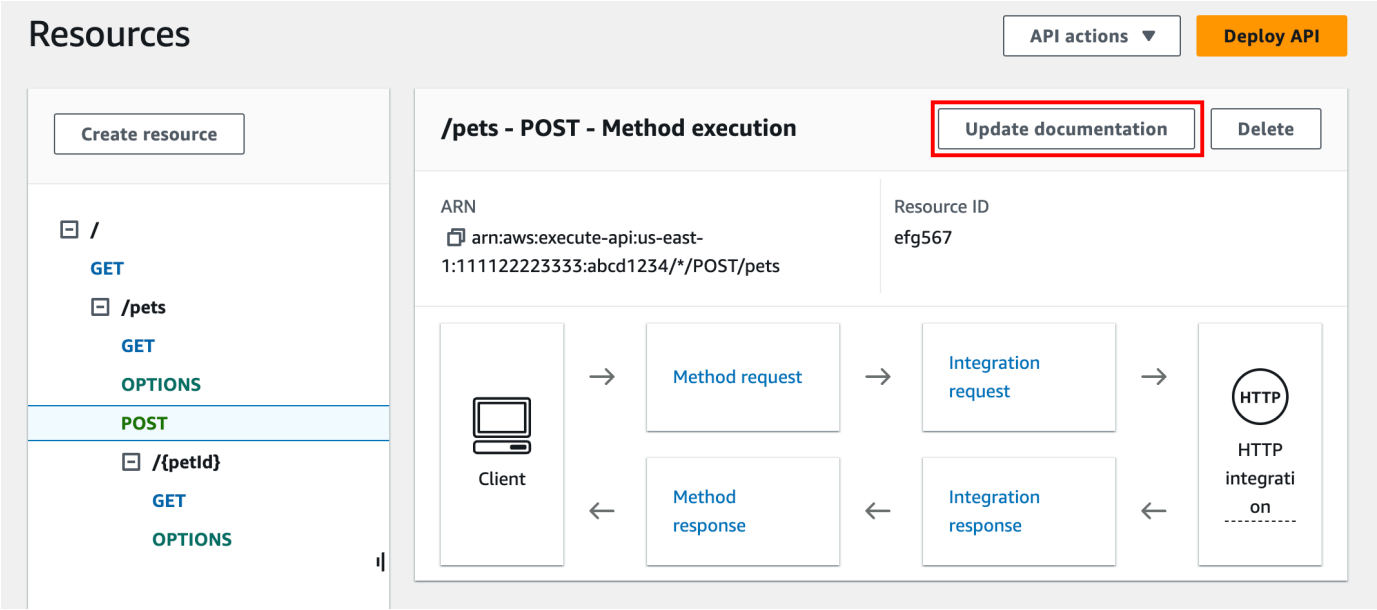
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie eine Beschreibung im Texteditor ein, zum Beispiel:

```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für eine nicht aufgeführte Methode erstellen.
7. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `METHOD`-Entität im Bereich Resources (Ressourcen) hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Resources (Ressourcen).
2. Wählen Sie die Methode aus und klicken Sie dann auf Update documentation (Dokumentation aktualisieren).



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können die Methode oder die Ressource auswählen, die die Methode enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`QUERY_PARAMETER`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `QUERY_PARAMETER`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Query parameter (Abfrageparameter) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Name einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführte Abfrageparameter erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Abfrageparameter oder die Ressource auswählen, die den Abfrageparameter enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **PATH_PARAMETER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine PATH_PARAMETER-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Path parameter (Pfadparameter) aus.
3. Geben Sie für Path (Pfad) einen Pfad ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Name einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführte Pfadparameter erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Pfadparameter oder die Ressource auswählen, die den Pfadparameter enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **REQUEST_HEADER**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine REQUEST_HEADER-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Request header (Anforderungsheader) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Anforderungsheader ein.

4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Name einen Namen ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Header erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Anforderungsheader oder die Ressource auswählen, die den Anforderungsheader enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`REQUEST_BODY`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `REQUEST_BODY`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Request body (Anforderungstext) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Anforderungstext ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie eine Beschreibung im Texteditor ein.
6. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Anforderungstext erstellen.
7. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Anforderungstext oder die Ressource auswählen, die den Anforderungstext enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`RESPONSE`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `RESPONSE`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen unter Documentation type (Dokumentationstyp) die Option Response (status code) (Antwort (Statuscode)) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für die Antwort ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antwort-Statuscode erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antwort-Statuscode oder die Ressource auswählen, die den Antwort-Statuscode enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`RESPONSE_HEADER`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `RESPONSE_HEADER`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Response header (Antwortheader) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Antwortheader ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.
6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antwortheader erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antwortheader oder die Ressource auswählen, die den Antwortheader enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`RESPONSE_BODY`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `RESPONSE_BODY`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Response body (Antworttext) aus.
3. Geben Sie unter Path (Pfad) einen Pfad für den Antworttext ein.
4. Wählen Sie ein HTTP-Verb für Method (Methode) aus.
5. Geben Sie unter Status code (Statuscode) einen HTTP-Statuscode ein.

6. Geben Sie eine Beschreibung im Texteditor ein.
7. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für einen nicht aufgeführten Antworttext erstellen.
8. Wiederholen Sie ggf. diese Schritte, um eine weitere Dokumentation hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Resources and methods (Ressourcen und Methoden) aus.
2. Sie können den Antworttext oder die Ressource auswählen, die den Antworttext enthält, und dann mithilfe der Suchleiste die Dokumentation suchen und auswählen.
3. Wählen Sie Bearbeiten aus.

Dokumentieren einer **`MODEL`**-Entität

Das Dokumentieren einer `MODEL`-Entität beinhaltet das Erstellen und Verwalten von `DocumentPart`-Instances für das Modell sowie der `properties` jedes Modells. Das Modell `Error`, das standardmäßig in jeder API enthalten ist, verfügt z. B. über die folgende Schemadefinition:

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

Es erfordert zwei `DocumentationPart`-Instances, eine für das `Model` und die andere für dessen `message`-Eigenschaft:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
```

```
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

und

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

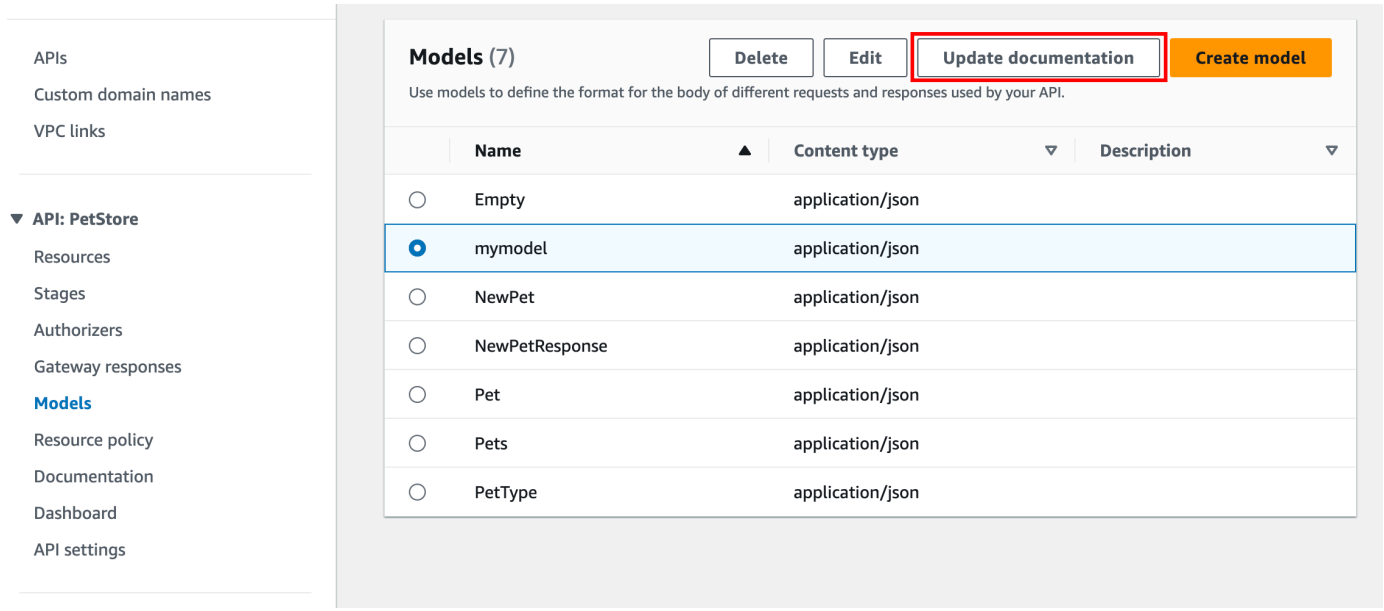
Wenn die API exportiert wird, überschreiben die Eigenschaften des `DocumentationPart` die Werte im ursprünglichen Schema.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `MODEL`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Model (Modell) aus.
3. Geben Sie für Name einen Modellnamen ein.
4. Geben Sie eine Beschreibung im Texteditor ein.
5. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für nicht aufgeführte Modelle erstellen.
6. Wiederholen Sie ggf. diese Schritte, um einen Dokumentationsbaustein zu anderen Modellen hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `MODEL`-Entität im Bereich Models (Modelle) hinzuzufügen:

1. Klicken Sie im Navigationsbereich auf Models (Modelle).
2. Wählen Sie das Modell aus und klicken Sie dann auf Update documentation (Dokumentation aktualisieren).



Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Models (Modelle) aus.
2. Verwenden Sie die Suchleiste oder wählen Sie das Modell aus und klicken Sie dann auf Edit (Bearbeiten).

Dokumentieren einer **`AUTHORIZER`**-Entität

Gehen Sie wie folgt vor, um eine neue Dokumentation für eine `AUTHORIZER`-Entität hinzuzufügen:

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation) und anschließend auf Create Documentation Part (Dokumentation erstellen).
2. Wählen Sie als Documentation type (Dokumentationstyp) die Option Authorizer (Genehmiger) aus.
3. Geben Sie unter Name, einen Namen für den Genehmiger ein.
4. Geben Sie eine Beschreibung im Texteditor ein. Geben Sie einen Wert für das gültige `location`-Feld für den Genehmiger ein.
5. Wählen Sie Create documentation part (Dokumentation erstellen) aus. Sie können Dokumentation für nicht aufgeführte Genehmiger erstellen.
6. Wiederholen Sie ggf. diese Schritte, um einen Dokumentationsbaustein zu anderen Genehmigern hinzuzufügen oder zu bearbeiten.

Gehen Sie wie folgt vor, um eine vorhandene Dokumentation zu bearbeiten:

1. Wählen Sie im Bereich Documentation (Dokumentation) die Registerkarte Authorizers (Genehmiger) aus.
2. Verwenden Sie die Suchleiste oder wählen Sie den Genehmiger aus und klicken Sie dann auf Edit (Bearbeiten).

Veröffentlichen der API-Dokumentation mithilfe der API Gateway-Konsole

Im folgenden Verfahren wird beschrieben, wie Sie eine Dokumentationsversion veröffentlichen.

So veröffentlichen Sie eine Dokumentationsversion mit der API Gateway-Konsole

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation).
2. Wählen Sie Publish documentation (Dokumentation veröffentlichen) aus.
3. Konfigurieren der Veröffentlichung:
   a. Wählen Sie für Stufe eine Stufe aus.
   b. Geben Sie für Version eine Versionskennung ein, z. B. `1.0.0`.
   c. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
4. Wählen Sie Publish.

Sie können nun die veröffentlichte Dokumentation herunterladen, indem Sie die Dokumentation in eine externe OpenAPI-Datei exportieren. Weitere Informationen hierzu finden Sie unter the section called “Exportieren einer REST-API”.

Dokumentieren einer API mit der API Gateway-REST-API

In diesem Abschnitt wird beschrieben, wie Sie die Dokumentationsbausteine einer API mit der API Gateway-REST-API erstellen und verwalten.

Vor dem Erstellen und Bearbeiten der Dokumentation einer API erstellen Sie zunächst die API. In diesem Abschnitt verwenden wir die PetStoreAPI als Beispiel. Um eine API mit der API Gateway-Konsole zu erstellen, befolgen Sie die Anweisungen in Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels.

Themen

Dokumentieren der **API**-Entität

Um Dokumentation für eine API hinzuzufügen, fügen Sie eine DocumentationPartRessource für die API-Entität hinzu:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "API"
    },
    "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  ...
  "id": "s2e5xf",
  "location": {
    "path": null,
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "API"
  },
  "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Wenn der Dokumentationsbaustein bereits hinzugefügt wurde, wird eine `409 Conflict`-Antwort mit der Fehlermeldung `Documentation part already exists for the specified location: type 'API'."` zurückgegeben. In diesem Fall müssen Sie die Operation documentationpart:update aufrufen.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API
 Gateway.\"\n\t}\n}"
  } ]
}
```

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit der aktualisierten `DocumentationPart`-Instance in der Nutzlast zurückgegeben.

Dokumentieren einer **`RESOURCE`**-Entität

Um Dokumentation für die Root-Ressource einer API hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die auf die entsprechende Ressourcenressource ausgerichtet ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
       "type" : "RESOURCE",
    },
    "properties" : "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    }
  },
```

```
  "id": "p76vqo",
  "location": {
    "path": "/",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Wenn kein Ressourcenpfad angegeben ist, wird davon ausgegangen, dass die Ressource die Stammressource ist. Sie können `"path": "/"` zu `properties` hinzufügen, um diese Spezifikation explizit zu machen.

Um eine Dokumentation für eine untergeordnete Ressource einer API zu erstellen, fügen Sie eine DocumentationPartRessource hinzu, die auf die entsprechende Ressourcenressource ausgerichtet ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets"
    },
    "properties": "{\n\t\"description\" : \"A child resource under the root of
 PetStore.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Beispielsweise:

```
{
  "_links": {
    "curies": {
```

```
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    }
  },
  "id": "qcht86",
  "location": {
    "path": "/pets",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource under the root of PetStore.
\"\n}"
}
```

Um Dokumentation für eine untergeordnete Ressource hinzuzufügen, die durch einen Pfadparameter angegeben wird, fügen Sie eine DocumentationPartRessource hinzu, die für die Ressourcenressource bestimmt ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets/{petId}"
```

```
    },
    "properties": "{\n\t\"description\" : \"A child resource specified by the petId
 path parameter.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Beispielsweise:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    }
  },
  "id": "k6fpwb",
  "location": {
    "path": "/pets/{petId}",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource specified by the petId path
 parameter.\"\n}"
}
```

> **Note**
>
> Die DocumentationPartInstanz einer `RESOURCE` Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Dokumentieren einer **`METHOD`**-Entität

Um Dokumentation für eine Methode einer API hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die auf die entsprechende Methodenressource ausgerichtet ist:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "METHOD",
         "path" : "/pets",
         "method" : "GET"
    },
    "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
```

```
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
```

```
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Wenn in der vorherigen Anforderung das Feld `location.method` nicht angegeben ist, wird davon ausgegangen, dass die ANY-Methode verwendet wird, dargestellt durch ein Platzhalterzeichen `*`.

Um den Dokumentationsinhalt einer `METHOD`-Entität zu aktualisieren, rufen Sie den Vorgang documentationpart:update auf, der eine neue `properties`-Zuordnung bereitstellt:

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
  } ]
}
```

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit der aktualisierten `DocumentationPart`-Instance in der Nutzlast zurückgegeben. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
```

```
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
}
```

Dokumentieren einer **QUERY_PARAMETER**-Entität

Um Dokumentation für einen Anforderungsabfrageparameter hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die auf den QUERY_PARAMETER Typ ausgerichtet ist und die gültigen Felder path und enthältname.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "QUERY_PARAMETER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "page"
    },
    "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    }
  },
  "id": "h9ht5w",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "page",
    "statusCode": null,
    "type": "QUERY_PARAMETER"
  },
  "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Die `properties`-Map des Dokumentationsbausteins einer `QUERY_PARAMETER`-Entität kann von einer der untergeordneten `QUERY_PARAMETER`-Entitäten übernommen werden. Wenn Sie z. B. eine `treats`-Ressource nach `/pets/{petId}` hinzufügen, die `GET`-Methode für `/pets/{petId}/treats` aktivieren und den `page`-Abfrageparameter bereitstellen, übernimmt dieser untergeordnete Abfrageparameter die `DocumentationPart`-Map des `properties` von dem gleichnamigen Abfrageparameter der `GET /pets`-Methode, es sei denn, Sie fügen explizit eine `DocumentationPart`-Ressource zum `page`-Abfrageparameter der `GET /pets/{petId}/treats`-Methode hinzu.

Dokumentieren einer **PATH_PARAMETER**-Entität

Um Dokumentation für einen Pfadparameter hinzuzufügen, fügen Sie eine DocumentationPartRessource für die PATH_PARAMETER Entität hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "PATH_PARAMETER",
         "path" : "/pets/{petId}",
         "method" : "*",
         "name" : "petId"
    },
    "properties": "{\n\t\"description\" : \"The id of the pet to retrieve.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    }
```

```
  },
  "id": "ckpgog",
  "location": {
    "path": "/pets/{petId}",
    "method": "*",
    "name": "petId",
    "statusCode": null,
    "type": "PATH_PARAMETER"
  },
  "properties": "{\n  \"description\" : \"The id of the pet to retrieve\"\n}"
}
```

Dokumentieren einer **`REQUEST_BODY`**-Entität

Um Dokumentation für einen Anforderungstext hinzuzufügen, fügen Sie eine DocumentationPartRessource für den Anfragetext hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "REQUEST_BODY",
         "path" : "/pets",
         "method" : "POST"
    },
    "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
```

```
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    }
  },
  "id": "kgmfr1",
  "location": {
    "path": "/pets",
    "method": "POST",
    "name": null,
    "statusCode": null,
    "type": "REQUEST_BODY"
  },
  "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Dokumentieren einer **REQUEST_HEADER**-Entität

Um Dokumentation für einen Anforderungsheader hinzuzufügen, fügen Sie eine DocumentationPartRessource für den Anforderungsheader hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "REQUEST_HEADER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "x-my-token"
    },
```

```
    "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    }
  },
  "id": "h0m3uf",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "x-my-token",
    "statusCode": null,
    "type": "REQUEST_HEADER"
  },
  "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Dokumentieren einer **`RESPONSE`**-Entität

Um eine Dokumentation für eine Antwort auf einen Statuscode hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die auf die entsprechende MethodResponseRessource ausgerichtet ist.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location": {
      "path": "/",
      "method": "*",
      "name": null,
      "statusCode": "200",
      "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
    "_links": {
        "self": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        }
    },
    "id": "lattew",
    "location": {
        "path": "/",
        "method": "*",
        "name": null,
        "statusCode": "200",
        "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
```

```
}
```

Dokumentieren einer **`RESPONSE_HEADER`**-Entität

Um Dokumentation für einen Antwort-Header hinzuzufügen, fügen Sie eine DocumentationPartRessource für den Antwort-Header hinzu.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Zum Beispiel:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
```

```
    }
  },
  "id": "fev7j7",
  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
}
```

Die Dokumentation dieses `Content-Type`-Antwort-Headers ist die Standarddokumentation für die `Content-Type`-Header aller Antworten der API.

Dokumentieren einer **`AUTHORIZER`**-Entität

Um Dokumentation für einen API-Autorisierer hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die für den angegebenen Autorisierer bestimmt ist.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "AUTHORIZER",
         "name" : "myAuthorizer"
    },
    "properties": "{\n\t\"description\" : \"Authorizes invocations of configured
 methods.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Beispielsweise:

```
{
```

```
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    }
  },
  "id": "pw3qw3",
  "location": {
    "path": null,
    "method": null,
    "name": "myAuthorizer",
    "statusCode": null,
    "type": "AUTHORIZER"
  },
  "properties": "{\n\t\"description\" : \"Authorizes invocations of configured methods.
\"\n}"
}
```

> **Note**
>
> Die DocumentationPartInstanz einer `AUTHORIZER` Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Dokumentieren einer **`MODEL`**-Entität

Das Dokumentieren einer `MODEL`-Entität beinhaltet das Erstellen und Verwalten von `DocumentPart`-Instances für das Modell sowie der `properties` jedes Modells. Das Modell `Error`, das standardmäßig in jeder API enthalten ist, verfügt z. B. über die folgende Schemadefinition:

```
{
```

```
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

Es erfordert zwei `DocumentationPart`-Instances, eine für das `Model` und die andere für dessen `message`-Eigenschaft:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

und

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Wenn die API exportiert wird, überschreiben die Eigenschaften des `DocumentationPart` die Werte im ursprünglichen Schema.

Um Dokumentation für ein API-Modell hinzuzufügen, fügen Sie eine DocumentationPartRessource hinzu, die auf das angegebene Modell ausgerichtet ist.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
```

```
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "MODEL",
         "name" : "Pet"
    },
    "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Bei Erfolg gibt die Operation eine `201 Created`-Antwort zurück, die die neu erstellte `DocumentationPart`-Instance in der Nutzlast enthält. Beispielsweise:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    }
  },
  "id": "lkn4uq",
  "location": {
    "path": null,
    "method": null,
    "name": "Pet",
    "statusCode": null,
    "type": "MODEL"
```

```
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Wiederholen Sie denselben Schritt, um eine DocumentationPart Instanz für eine der Eigenschaften des Modells zu erstellen.

> **Note**
>
> Die DocumentationPartInstanz einer `MODEL` Entität kann von keiner ihrer untergeordneten Ressourcen vererbt werden.

Aktualisieren von Dokumentationsbausteinen

Um die Dokumentationsteile jeder Art von API-Entitäten zu aktualisieren, reichen Sie eine PATCH-Anfrage für eine DocumentationPartInstanz einer bestimmten Teil-ID ein, um die vorhandene `properties` Map durch eine neue zu ersetzen.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}
```

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit der aktualisierten `DocumentationPart`-Instance in der Nutzlast zurückgegeben.

Sie können mehrere Dokumentationsbausteine in einer einzigen `PATCH`-Anforderung aktualisieren.

Auflisten von Dokumentationsbausteinen

Um die Dokumentationsteile jeder Art von API-Entitäten aufzulisten, reichen Sie eine GET-Anfrage für eine DocumentationPartsSammlung ein.

```
GET /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Bei einer erfolgreichen Antwort wird der Statuscode `200 OK` sowie eine Nutzlast mit den verfügbaren `DocumentationPart`-Instances in der Nutzlast zurückgegeben.

Veröffentlichen der API-Dokumentation mithilfe der API Gateway-REST-API

Um die Dokumentation für eine API zu veröffentlichen, erstellen oder aktualisieren Sie einen Dokumentations-Snapshot bzw. rufen Sie einen Dokumentations-Snapshot ab und verknüpfen Sie diesen mit einer API-Stufe. Beim Erstellen eines Dokumentations-Snapshots können Sie ihn gleichzeitig mit einer eine API-Stufe verknüpfen.

Themen

- Erstellen eines Dokumentations-Snapshots und dessen Verknüpfen mit einer API-Stufe
- Erstellen eines Dokumentations-Snapshots
- Aktualisieren eines Dokumentations-Snapshots
- Abrufen eines Dokumentations-Snapshots
- Verknüpfen eines Dokumentations-Snapshots mit einer API-Stufe
- Herunterladen eines mit einer Stufe verknüpften Dokumentations-Snapshots

Erstellen eines Dokumentations-Snapshots und dessen Verknüpfen mit einer API-Stufe

Um einen Snapshot der API-Dokumentationsbausteine zu erstellen und ihn zugleich mit einer API-Stufe zu verknüpfen, übermitteln Sie die folgende `POST`-Anforderung:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
```

```
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "stageName": "prod",
    "description" : "My API Documentation v1.0.0"
}
```

Bei Erfolg gibt die Operation eine `200 OK`-Antwort zurück, die die neu erstellte `DocumentationVersion`-Instance als Nutzlast enthält.

Sie können auch einen Dokumentations-Snapshot erstellen, ohne ihn einer API-Stufe zuzuweisen, und dann restapi:update aufrufen, um den Snapshot mit einer bestimmten API-Stufe zu verknüpfen. Sie können auch einen vorhandene Dokumentations-Snapshot aktualisieren oder abfragen und dann dessen Verknüpfung ändern. Diese Schritte werden in den nächsten vier Abschnitten gezeigt.

Erstellen eines Dokumentations-Snapshots

Um einen Snapshot der Dokumentationsteile einer API zu erstellen, erstellen Sie eine neue DocumentationVersionRessource und fügen Sie sie der DocumentationVersionsSammlung der API hinzu:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "description" : "My API Documentation v1.0.0"
}
```

Bei Erfolg gibt die Operation eine `200 OK`-Antwort zurück, die die neu erstellte `DocumentationVersion`-Instance als Nutzlast enthält.

Aktualisieren eines Dokumentations-Snapshots

Sie können einen Dokumentations-Snapshot nur aktualisieren, indem Sie die `description` Eigenschaft der entsprechenden DocumentationVersionRessource ändern. Im folgenden Beispiel wird gezeigt, wie Sie die Beschreibung des Dokumentations-Snapshots gemäß seiner Versionskennung, *`version`*, aktualisieren (z. B. `1.0.0`).

```
PATCH /restapis/restapi_id/documentation/versions/version HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/description",
        "value": "My API for testing purposes."
    }]
}
```

Bei Erfolg gibt die Operation eine `200 OK`-Antwort zurück, die die aktualisierte `DocumentationVersion`-Instance als Nutzlast enthält.

Abrufen eines Dokumentations-Snapshots

Um eine Momentaufnahme der Dokumentation zu erhalten, reichen Sie eine `GET` Anfrage für die angegebene DocumentationVersionRessource ein. Im folgenden Beispiel wird gezeigt, wie Sie einen Dokumentations-Snapshot für eine bestimmte Versionskennung (1.0.0) abrufen.

```
GET /restapis/<restapi_id>/documentation/versions/1.0.0 HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Verknüpfen eines Dokumentations-Snapshots mit einer API-Stufe

Um die API-Dokumentation zu veröffentlichen, verknüpfen Sie einen Dokumentations-Snapshot mit einer API-Stufe. Sie müssen bereits eine API-Stufe erstellt haben, bevor Sie die Dokumentationsversion damit verknüpfen können.

Um einen Dokumentations-Snapshot mit einer API-Stufe über die API Gateway-REST-API zu verknüpfen, rufen Sie den Vorgang stage:update auf, um die gewünschte Dokumentationsversion für die Eigenschaft `stage.documentationVersion` festzulegen:

```
PATCH /restapis/RESTAPI_ID/stages/STAGE_NAME
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/documentationVersion",
        "value": "VERSION_IDENTIFIER"
    }]
}
```

Herunterladen eines mit einer Stufe verknüpften Dokumentations-Snapshots

Nachdem eine Version der Dokumentationsbausteine mit einer Stufe verknüpft wurde, können Sie die Dokumentationsbausteine zusammen mit den API-Entitätsdefinitionen mithilfe der API-Gateway-Konsole, der API-Gateway-REST-API, eines der zugehörigen SDKs oder der AWS CLI für API Gateway in eine externe Datei exportieren. Der Vorgang ist mit dem für das Exportieren der API identisch. Das exportierte Dateiformat kann JSON oder YAML sein.

Mit der API Gateway-REST-API können Sie auch explizit den Abfrageparameter `extension=documentation,integrations,authorizers` so konfigurieren, dass die API-Dokumentationsbausteine, API-Integrationen und Genehmiger beim API-Export berücksichtigt werden. Standardmäßig sind die Dokumentationsbausteine enthalten, die Integrationen und Genehmiger aber vom Export einer API ausgeschlossen. Die Standardausgabe von einem API-Export eignet sich für die Verteilung der Dokumentation.

Um die API-Dokumentation in eine externe JSON-OpenAPI-Datei mit der API Gateway-REST-API zu exportieren, übermitteln Sie die folgende GET-Anforderung:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
 HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Hier enthält das Objekt `x-amazon-apigateway-documentation` die Dokumentationsbausteine und die API-Entitätsdefinitionen beinhalten die von OpenAPI unterstützten Dokumentationseigenschaften. Die Ausgabe enthält keine Details zur Integration oder zu Lambda-Genehmigern (ehemals als benutzerdefinierte Genehmiger bezeichnet). Um beide Details einzuschließen, konfigurieren Sie `extensions=integrations,authorizers,documentation`. Um Details zu Integrationen, aber nicht zu Genehmigern einzuschließen, konfigurieren Sie `extensions=integrations,documentation`.

Sie müssen den `Accept:application/json`-Header in der Anforderung so konfigurieren, dass die Ausgabe in einer JSON-Datei erfolgt. Um eine YAML-Ausgabe zu erzeugen, ändern den Header der Anforderung in `Accept:application/yaml`.

Als Beispiel wird eine API verwendet, die eine einfache `GET`-Methode für die Stammressource (`/`) bereitstellt. Diese API verfügt über vier API-Entitäten, die in einer OpenAPI-Definitionsdatei definiert sind, eine für jeden der Typen `API`, `MODEL`, `METHOD` und `RESPONSE`. Jeder der `API`-, `METHOD`- und `RESPONSE`-Entitäten wurde ein Dokumentationsbaustein hinzugefügt. Durch Aufrufen des vorherigen Befehls zum Exportieren der Dokumentation erhalten wir die folgende Ausgabe, wobei die Dokumentationsbausteine im Objekt `x-amazon-apigateway-documentation` als Erweiterung einer OpenAPI-Standarddatei aufgeführt werden.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "description": "API info description",
      "version": "2016-11-22T22:39:14Z",
```

```
      "title": "doc",
      "x-bar": "API info x-bar"
   },
   "paths": {
      "/": {
         "get": {
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            },
            "x-example": "x- Method example"
         },
         "x-bar": "resource x-bar"
      }
   },
   "x-amazon-apigateway-documentation": {
      "version": "1.0.0",
      "createdDate": "2016-11-22T22:41:40Z",
      "documentationParts": [
         {
            "location": {
               "type": "API"
            },
            "properties": {
               "description": "API description",
               "foo": "API foo",
               "x-bar": "API x-bar",
               "info": {
                  "description": "API info description",
                  "version": "API info version",
                  "foo": "API info foo",
                  "x-bar": "API info x-bar"
               }
            }
         },
```

```
         {
            "location": {
               "type": "METHOD",
               "method": "GET"
            },
            "properties": {
               "description": "Method description.",
               "x-example": "x- Method example",
               "foo": "Method foo",
               "info": {
                  "version": "method info version",
                  "description": "method info description",
                  "foo": "method info foo"
               }
            }
         },
         {
            "location": {
               "type": "RESOURCE"
            },
            "properties": {
               "description": "resource description",
               "foo": "resource foo",
               "x-bar": "resource x-bar",
               "info": {
                  "description": "resource info description",
                  "version": "resource info version",
                  "foo": "resource info foo",
                  "x-bar": "resource info x-bar"
               }
            }
         }
      ]
   },
   "x-bar": "API x-bar",
   "servers": [
      {
         "url": "https://rznaap68yi.execute-api.ap-southeast-1.amazonaws.com/
{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
            }
         }
```

```
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "description" : "API info description",
    "version" : "2016-11-22T22:39:14Z",
    "title" : "doc",
    "x-bar" : "API info x-bar"
  },
  "host" : "rznaap68yi.execute-api.ap-southeast-1.amazonaws.com",
  "basePath" : "/test",
  "schemes" : [ "https" ],
  "paths" : {
    "/" : {
      "get" : {
        "description" : "Method description.",
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            }
          }
        },
        "x-example" : "x- Method example"
      },
      "x-bar" : "resource x-bar"
    }
  },
```

```
  "definitions" : {
    "Empty" : {
      "type" : "object",
      "title" : "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation" : {
    "version" : "1.0.0",
    "createdDate" : "2016-11-22T22:41:40Z",
    "documentationParts" : [ {
      "location" : {
        "type" : "API"
      },
      "properties" : {
        "description" : "API description",
        "foo" : "API foo",
        "x-bar" : "API x-bar",
        "info" : {
          "description" : "API info description",
          "version" : "API info version",
          "foo" : "API info foo",
          "x-bar" : "API info x-bar"
        }
      }
    }, {
      "location" : {
        "type" : "METHOD",
        "method" : "GET"
      },
      "properties" : {
        "description" : "Method description.",
        "x-example" : "x- Method example",
        "foo" : "Method foo",
        "info" : {
          "version" : "method info version",
          "description" : "method info description",
          "foo" : "method info foo"
        }
      }
    }, {
      "location" : {
        "type" : "RESOURCE"
      },
      "properties" : {
```

```
        "description" : "resource description",
        "foo" : "resource foo",
        "x-bar" : "resource x-bar",
        "info" : {
          "description" : "resource info description",
          "version" : "resource info version",
          "foo" : "resource info foo",
          "x-bar" : "resource info x-bar"
        }
      }
    } ]
  },
  "x-bar" : "API x-bar"
}
```

Bei einem OpenAPI-kompatiblen Attribut, das in der `properties`-Map eines Dokumentationsbausteins definiert ist, fügt API Gateway das Attribut in die zugehörige API-Entitätsdefinition ein. Das Attribut `x-`*`something`* ist eine OpenAPI-Standarderweiterung. Diese Erweiterung wird in die API-Entitätsdefinition propagiert. Schauen Sie sich z. B. das Attribut `x-example` für die `GET`-Methode an. Ein Attribut wie `foo` ist nicht Teil der OpenAPI-Spezifikation und wird nicht in die dazugehörigen API-Entitätsdefinitionen eingefügt.

Wenn ein Dokumentations-Rendering-Tool (z. B. OpenAPI UI) die API-Entitätsdefinitionen analysiert, um Dokumentationsattribute zu extrahieren, sind alle nicht mit OpenAPI kompatiblen `properties`-Attribute einer `DocumentationPart`-Instance nicht für das Tool verfügbar. Wenn jedoch ein Dokumentations-Rendering-Tool das Objekt `x-amazon-apigateway-documentation` analysiert, um Inhalte zu erhalten, oder wenn das Tool restapi:documentation-parts und documenationpart:by-id aufruft, um Dokumentationsbausteine von API Gateway abzurufen, sind alle Dokumentationsattribute für das Tool zur Anzeige verfügbar.

Um die Dokumentation mit API-Entitätsdefinitionen, die Integrationsdetails enthalten, in eine JSON-OpenAPI-Datei zu exportieren, übermitteln Sie folgende `GET`-Anforderung:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,documentation HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Um die Dokumentation mit API-Entitätsdefinitionen, die Details zu Integrationen und Genehmigern enthalten, in eine YAML-OpenAPI-Datei zu exportieren, übermitteln Sie die folgende GET-Anforderung:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,authorizers,documentation HTTP/1.1
Accept: application/yaml
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Um die API Gateway-Konsole zum Exportieren und Herunterladen der veröffentlichten Dokumentation einer API zu verwenden, befolgen Sie die Anweisungen unter REST-API über die API-Gateway-Konsole exportieren.

Importieren einer API-Dokumentation

Wie beim Importieren von API-Entitätsdefinitionen können Sie Dokumentationsbausteine von einer externen OpenAPI-Datei in eine API in API Gateway importieren. Sie geben die to-be-imported Dokumentationsteile innerhalb der x-amazon-apigateway-documentation Objekt Erweiterung in einer gültigen OpenAPI-Definitionsdatei an. Durch das Importieren der Dokumentation werden die vorhandenen API-Entitätsdefinitionen nicht geändert.

Sie haben die Möglichkeit, die neu angegebenen Dokumentationsbausteine in vorhandene Dokumentationsbausteine in API Gateway zusammenzuführen oder die vorhandenen Dokumentationsbausteine zu überschreiben. Im `MERGE`-Modus wird ein neuer Dokumentationsbaustein, der in der OpenAPI-Datei definiert ist, der `DocumentationParts`-Sammlung der API hinzugefügt. Wenn bereits ein importierter `DocumentationPart` vorhanden ist, ersetzt ein importiertes Attribut jeweils einen vorhandenen Baustein, sofern einer der beiden unterschiedlich ist. Andere vorhandene Dokumentationsattribute bleiben davon unberührt. Im Modus `OVERWRITE` wird die gesamte `DocumentationParts`-Sammlung entsprechend der importierten OpenAPI-Definitionsdatei ersetzt.

Importieren der Dokumentationsbausteine mit der API Gateway-REST-API

Um eine API-Dokumentation mit der API Gateway-REST-API zu importieren, rufen Sie den Vorgang documentationpart:import auf. Das folgende Beispiel zeigt, wie vorhandene Dokumentationsbausteine einer API mit einer einzigen `GET /` -Methode überschrieben werden. Bei Erfolg wird eine `200 OK`-Antwort zurückgegeben.

OpenAPI 3.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
   "openapi": "3.0.0",
   "info": {
      "description": "description",
      "version": "1",
      "title": "doc"
   },
   "paths": {
      "/": {
         "get": {
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            }
         }
      }
   },
```

```
  "x-amazon-apigateway-documentation": {
     "version": "1.0.3",
     "documentationParts": [
        {
           "location": {
              "type": "API"
           },
           "properties": {
              "description": "API description",
              "info": {
                 "description": "API info description 4",
                 "version": "API info version 3"
              }
           }
        },
        {
           "location": {
              "type": "METHOD",
              "method": "GET"
           },
           "properties": {
              "description": "Method description."
           }
        },
        {
           "location": {
              "type": "MODEL",
              "name": "Empty"
           },
           "properties": {
              "title": "Empty Schema"
           }
        },
        {
           "location": {
              "type": "RESPONSE",
              "method": "GET",
              "statusCode": "200"
           },
           "properties": {
              "description": "200 response"
           }
        }
     ]
```

```
   },
   "servers": [
      {
         "url": "/"
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "swagger": "2.0",
  "info": {
    "description": "description",
    "version": "1",
    "title": "doc"
  },
  "host": "",
  "basePath": "/",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "description": "Method description.",
        "produces": [
```

```
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
      {
        "location": {
          "type": "API"
        },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
        "location": {
          "type": "METHOD",
          "method": "GET"
        },
        "properties": {
          "description": "Method description."
        }
      },
      {
```

```
        "location": {
          "type": "MODEL",
          "name": "Empty"
        },
        "properties": {
          "title": "Empty Schema"
        }
      },
      {
        "location": {
          "type": "RESPONSE",
          "method": "GET",
          "statusCode": "200"
        },
        "properties": {
          "description": "200 response"
        }
      }
    ]
  }
}
```

Bei Erfolg gibt diese Anforderung eine "200 OK"-Antwort zurück, die die importierte `DocumentationPartId` in der Nutzlast enthält.

```
{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}
```

Sie können auch restapi:import oder restapi:put aufrufen. Die Dokumentationsbausteine im Objekt `x-amazon-apigateway-documentation` werden dann als Teil der OpenAPI-Eingabedatei der API-Definition bereitgestellt. Um die Dokumentationsbausteine vom API-Import auszuschließen, konfigurieren Sie `ignore=documentation` in den Abfrageparametern der Anforderung.

Importieren der Dokumentationsbausteine mit der API Gateway-Konsole

In den folgenden Anweisungen wird beschrieben, wie Sie Dokumentationsbausteine importieren.

So verwenden Sie die Konsole, um Dokumentationsbausteine einer API aus einer externen Datei zu importieren

1. Klicken Sie im Hauptnavigationsbereich auf Documentation (Dokumentation).
2. Wählen Sie Importieren aus.
3. Wenn Sie bereits über eine Dokumentation verfügen, wählen Sie Overwrite (Überschreiben) oder Merge (Zusammenführen) aus, um Ihre neue Dokumentation entweder zu überschreiben oder zusammenzuführen.
4. Wählen Sie Choose file (Datei auswählen) aus, um eine Datei von einem Laufwerk zu laden, oder geben Sie die Inhalte einer Datei in die Dateiansicht ein. Ein Beispiel finden Sie in der Nutzlast der Beispielanforderung unter Importieren der Dokumentationsbausteine mit der API Gateway-REST-API.
5. Wählen Sie aus, wie mit Warnungen beim Import umgegangen werden soll. Wählen Sie entweder Fail on warnings (Bei Warnungen fehlschlagen) oder Ignore warnings (Warnungen ignorieren). Weitere Informationen finden Sie unter the section called “Fehler und Warnungen beim Import”.
6. Wählen Sie Import.

Steuern des Zugriffs auf Ihre API-Dokumentation

Wenn Sie über ein eigenes Dokumentationsteam verfügen, das Ihre API-Dokumentation verfasst und bearbeitet, können Sie separate Zugriffsberechtigungen für Ihre Entwickler (für die API-Entwicklung) und für Ihre Autoren und Editoren (für die Entwicklung von Inhalten) konfigurieren. Dies ist besonders dann sinnvoll, wenn ein Drittanbieter an der Erstellung der Dokumentation beteiligt ist.

Um Ihrem Dokumentationsteam Zugriff auf die Erstellung, Aktualisierung und Veröffentlichung Ihrer API-Dokumentation zu gewähren, können Sie dem Dokumentationsteam eine IAM-Rolle mit der folgenden IAM-Richtlinie zuweisen, wobei *account_id die AWS Konto-ID* Ihres Dokumentationsteams ist.

```
{
  "Version": "2012-10-17",
  "Statement": [

    {
      "Sid": "StmtDocPartsAddEditViewDelete",
      "Effect": "Allow",
      "Action": [
```

```
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway::account_id:/restapis/*/documentation/*"
      ]
    }
  ]
}
```

Weitere Informationen zum Konfigurieren von Berechtigungen für den Zugriff auf API Gateway-Ressourcen finden Sie unter the section called “Funktionsweise von Amazon API Gateway mit IAM”.

Generieren eines SDK für eine REST-API in API Gateway

Um Ihre REST-API auf plattform- oder sprachenspezifische Weise aufzurufen, müssen Sie das plattform- oder sprachenspezifische SDK der API generieren. Sie generieren Ihr SDK, nachdem Sie Ihre API erstellt, getestet und in einer Phase bereitgestellt haben. Derzeit unterstützt API Gateway die Generierung eines SDK für eine API in Java, Java für Android JavaScript, Objective-C oder Swift für iOS und Ruby.

In diesem Abschnitt wird erläutert, wie ein SDK einer API Gateway-API generiert wird. Es zeigt auch, wie das generierte SDK in einer Java-App, einer Java-App für Android, Objective-C- und Swift für iOS-Apps und einer JavaScript App verwendet wird.

Zur Vereinfachung der Erläuterung verwenden wir die API Gateway-API, die diese einfache Lambda-Rechnerfunktion bereitstellt.

Bevor Sie fortfahren, erstellen oder importieren Sie die API und stellen sie mindestens einmal in API Gateway bereit. Detaillierte Anweisungen finden Sie unter Bereitstellen einer REST-API in Amazon API Gateway.

Themen

- Einfache Lambda-Rechnerfunktion
- Einfache Rechner-API in API Gateway
- OpenAPI-Definition für einfache Rechner-API
- Generieren des Java SDK einer API

- Generieren des Android SDKs einer API
- Generieren des iOS SDKs einer API
- Generieren Sie das JavaScript SDK einer REST-API
- Generieren des Ruby SDK einer API
- Generieren Sie SDKs für eine API mithilfe von Befehlen AWS CLI

Einfache Lambda-Rechnerfunktion

Zur Veranschaulichung verwenden wir eine Node.js-Lambda-Funktion, die die binären Vorgänge Addition, Subtraktion, Multiplikation und Division durchführt.

Themen

- Eingabeformat der einfachen Lambda-Rechnerfunktion
- Ausgabeformat der einfachen Lambda-Rechnerfunktion
- Implementieren der einfachen Lambda-Rechnerfunktion

Eingabeformat der einfachen Lambda-Rechnerfunktion

Diese Funktion nimm eine Eingabe im folgenden Format entgegen:

```
{ "a": "Number", "b": "Number", "op": "string"}
```

wobei op für jeden der Werte `(+, -, *, /, add, sub, mul, div)` stehen kann.

Ausgabeformat der einfachen Lambda-Rechnerfunktion

Wenn ein Vorgang erfolgreich ist, wird das Ergebnis im folgenden Format zurückgegeben:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number"}
```

wobei `c` das Ergebnis der Berechnung enthält.

Implementieren der einfachen Lambda-Rechnerfunktion

Die Implementierung der Lambda-Funktion wird wie folgt durchgeführt:

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

```
  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
    return "400 Invalid Input";
  }

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

  return res;
};
```

Einfache Rechner-API in API Gateway

Unsere einfache Rechner-API stellt drei Methoden (GET, POST, GET) zum Aufrufen der berei the section called “Einfache Lambda-Rechnerfunktion”. Eine grafische Darstellung dieser API sieht wie folgt aus:


Resources
Create resource
/
GET
POST
/{a}
ANY
/{b}
ANY
/{op}
GET

Diese drei Methoden zeigen verschiedene Methoden zur Bereitstellung der Eingabe für die Backend-Lambda-Funktion, um denselben Vorgang auszuführen:

- Die `GET /?a=...&b=...&op=...`-Methode verwendet die Abfrageparameter, um den Eingang anzugeben.
- Die `POST /`-Methode verwendet die JSON-Nutzlast `{"a":"Number", "b":"Number", "op":"string"}`, um den Eingang anzugeben.
- Die `GET /{a}/{b}/{op}`-Methode verwendet die Pfadparameter, um den Eingang anzugeben.

Ist keine Definition vorhanden, generiert API Gateway den entsprechenden SDK-Methodennamen durch eine Kombination der HTTP-Methode und den Pfadbestandteilen. Der Root-Pfadbestandteil (`/`) wird als `Api Root` bezeichnet. Beispiel: Der Standardname der Java-SDK-Methoden für die API-Methode von `GET /?a=...&b=...&op=...` lautet `getAB0p`, der Standardname der SDK-Methode für `POST /` lautet `postApiRoot` und der Standardname der SDK-Methode für `GET /{a}/{b}/{op}` lautet `getAB0p`. Einzelne SDKs können die Konvention anpassen. Weitere Informationen finden Sie in der Dokumentation in der generierten SDK-Quelle für SDK-spezifische Methodennamen.

Sie können, und sollten, die Standardnamen der SDK-Methode überschreiben, indem Sie die Eigenschaft operationName für jede API-Methode angeben. Dies ist bei Erstellung der API-Methode oder der Aktualisierung der API-Methode mit der API Gateway-REST-API möglich. In der API-Swagger-Definition können Sie die `operationId` festlegen, um dasselbe Ergebnis zu erzielen.

Bevor wir zeigen, wie diese Methoden mit einem SDK aufgerufen werden, das von API Gateway für diese API generiert wurde, fassen wir die Einrichtung kurz zusammen. Detaillierte Anweisungen finden Sie unter REST-API in API Gateway entwickeln. Wenn API Gateway für Sie neu ist, lesen Sie zuerst Wählen Sie ein AWS Lambda Integrations-Tutorial.

Erstellen von Modellen für Eingaben und Ausgaben

Wir müssen ein `Input`-Modell für die API erstellen, um stark typisierte Eingaben im SDK anzugeben. Für die Beschreibung des Antworttextdatentyps erstellen wir ein `Output`-Modell und ein `Result`-Modell.

Erstellen von Modellen für Eingabe, Ausgabe und Ergebnis

1. Klicken Sie im Navigationsbereich auf Models (Modelle).
2. Wählen Sie Modell erstellen aus.
3. Geben Sie unter Name **`input`** ein.

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

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

5. Geben Sie für Modellschema Folgendes ein:

```
{
    "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"Input"
}
```

6. Wählen Sie Modell erstellen aus.

7. Wiederholen Sie die folgenden Schritte, um jeweils ein `Output`- und ein `Result`-Modell zu erstellen.

   Geben Sie für das `Output`-Modell Folgendes für Modellschema ein:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "c": {"type":"number"}
    },
    "title": "Output"
}
```

   Geben Sie für das `Result`-Modell Folgendes für Modellschema ein: Ersetzen Sie die API-ID `abc123` durch Ihre API-ID.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
```

```
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Input"
        },
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Output"
        }
    },
    "title":"Result"
}
```

Einrichten von GET-/Abfrageparameter-Methode

Bei der `GET /?a=..&b=..&op=..`-Methode werden die Abfrageparameter in Method Request angegeben:

Einrichten der GET/URL-Abfragezeichenfolgenparameter

1. Klicken Sie im Abschnitt Methodenanforderung für die `GET`-Methode in der Stamm-Ressource (`/`) auf Bearbeiten.
2. Wählen Sie URL query string parameters (URL-Abfragezeichenfolgen-Parameter) und gehen Sie wie folgt vor:

   a. Wählen Sie Abfragezeichenfolge hinzufügen aus.

   b. Geben Sie unter Name **a** ein.

   c. Lassen Sie die Optionen Required (Obligatorisch) and Caching (Cache) deaktiviert.

   d. Caching bleibt ausgeschaltet.

   Wiederholen Sie die obigen Schritte und erstellen Sie eine Abfragezeichenfolge mit dem Namen **b** sowie eine Abfragezeichenfolge mit dem Namen **op**.
3. Wählen Sie Speichern.

Einrichten eines Datenmodells für die Nutzlast als Eingabe für das Backend

Bei der `POST /`-Methode erstellen wir das `Input`-Modell und fügen Sie es der Methodenabfrage hinzu, um die Form der Eingabedaten zu definieren.

Einrichten des Datenmodells für die Nutzlast als Eingabe für das Backend

1. Klicken Sie im Abschnitt Methodenanforderung für die `POST`-Methode in der Stamm-Ressource (`/`) auf Bearbeiten.
2. Klicken Sie auf Anforderungstext.
3. Wählen Sie Add model aus.
4. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
5. Wählen Sie für Modell die Option Eingabe aus.
6. Wählen Sie Speichern.

Bei diesem Modell können Ihre API-Kunden das SDK aufrufen, um die Eingabe eines `Input`-Objekts zu instanziieren. Ohne dieses Modell müssten die Kunden ein Dictionary-Objekt erstellen, um die JSON-Eingabe in der Lambda-Funktion darzustellen.

Einrichten eines Datenmodells für die Ergebnisausgabe vom Backend

Für alle drei Methoden erstellen wir das `Result`-Modell und fügen es zur `Method Response` der Methode hinzu, um die Form der Ausgabe zu definieren, die von der Lambda-Funktion zurückgegeben wird.

Einrichten des Datenmodells für die Ergebnisausgabe aus dem Backend

1. Wählen Sie die Ressource /{a}/{b}/{op} und dann die GET-Methode aus.
2. Klicken Sie auf der Registerkarte Methodenantwort unter Antwort 200 auf Bearbeiten.
3. Klicken Sie unter Anforderungstext auf Modell hinzufügen.
4. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
5. Wählen Sie für Modell die Option Ergebnis aus.
6. Wählen Sie Speichern.

Bei diesem Modell können Ihre API-Kunden eine erfolgreiche Ausgabe parsen, indem sie die Eigenschaften für ein `Result`-Objekt lesen. Ohne dieses Modell müssten die Kunden ein Dictionary-Objekt erstellen, um die JSON-Ausgabe darzustellen.

OpenAPI-Definition für einfache Rechner-API

Nachfolgend finden Sie die OpenAPI-Definition für eine einfache Rechner-API. Diese können Sie in Ihr Konto importieren. Nach dem Import müssen jedoch die ressourcenbasierten Berechtigungen für die Lambda-Funktion zurückgesetzt werden. Dazu wählen Sie die für Ihr Konto erstellte Lambda-Funktion in der API Gateway-Konsole unter Integration Request (Integrationsanforderung) erneut aus. Dann setzt die API Gateway-Konsole die erforderlichen Berechtigungen zurück. Alternativ können Sie die AWS Command Line Interface für den Lambda-Befehl add-permission verwenden.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-29T20:27:30Z",
    "title": "SimpleCalc"
  },
  "host": "t6dve4zn25.execute-api.us-west-2.amazonaws.com",
  "basePath": "/demo",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "op",
            "in": "query",
            "required": false,
            "type": "string"
          },
          {
            "name": "a",
            "in": "query",
            "required": false,
            "type": "string"
```

```
          },
          {
            "name": "b",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
```

```
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Input"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      }
    },
    "/{a}": {
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
```

```
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
            "description": "404 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
    "/{a}/{b}": {
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
```

```
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
            "description": "404 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
    "/{a}/{b}/{op}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
```

```
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "op",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
```

```
              }
            }
          },
          "type": "aws"
        }
      }
    }
  },
  "definitions": {
    "Input": {
      "type": "object",
      "properties": {
        "a": {
          "type": "number"
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
          "$ref": "#/definitions/Output"
        }
      },
      "title": "Result"
```

```
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "SimpleCalc",
    "version" : "2016-09-29T20:27:30Z"
  },
  "servers" : [ {
    "url" : "https://t6dve4zn25.execute-api.us-west-2.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "demo"
      }
    }
  } ],
  "paths" : {
    "/{a}/{b}" : {
      "x-amazon-apigateway-any-method" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "404" : {
            "description" : "404 response",
            "content" : { }
          }
```

```
        },
        "x-amazon-apigateway-integration" : {
          "type" : "mock",
          "responses" : {
            "default" : {
              "statusCode" : "404",
              "responseTemplates" : {
                "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    },
    "/{a}/{b}/{op}" : {
      "get" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "op",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
```

```
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "passthroughBehavior" : "when_no_templates"
        }
      }
    },
    "/" : {
      "get" : {
        "parameters" : [ {
          "name" : "op",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
```

```
          "name" : "a",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
```

```
          "passthroughBehavior" : "when_no_templates"
        }
      },
      "post" : {
        "requestBody" : {
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Input"
              }
            }
          },
          "required" : true
        },
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
```

```
    },
    "/{a}" : {
      "x-amazon-apigateway-any-method" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "404" : {
            "description" : "404 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "mock",
          "responses" : {
            "default" : {
              "statusCode" : "404",
              "responseTemplates" : {
                "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Input" : {
        "title" : "Input",
        "type" : "object",
        "properties" : {
          "a" : {
            "type" : "number"
```

```
          },
          "b" : {
            "type" : "number"
          },
          "op" : {
            "type" : "string"
          }
        }
      },
      "Output" : {
        "title" : "Output",
        "type" : "object",
        "properties" : {
          "c" : {
            "type" : "number"
          }
        }
      },
      "Result" : {
        "title" : "Result",
        "type" : "object",
        "properties" : {
          "input" : {
            "$ref" : "#/components/schemas/Input"
          },
          "output" : {
            "$ref" : "#/components/schemas/Output"
          }
        }
      }
    }
  }
}
```

Generieren des Java SDK einer API

So generieren Sie das Java SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.

3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Java-Plattform aus und gehen Sie wie folgt vor:

   a. Geben Sie für Service Name den Namen Ihres SDK an. Beispiel, **`SimpleCalcSdk`**. Dies wird der Name Ihrer SDK-Client-Klasse. Der Name entspricht dem `<name>`-Tag unter `<project>` in der pom.xml-Datei, die sich im Projektordner des SDK befindet. Verwenden Sie keine Bindestriche.

   b. Geben Sie für Java Package Name einen Paketnamen für Ihr SDK an. Beispiel, **`examples.aws.apig.simpleCalc.sdk`**. Dieser Paketname wird als Namespace Ihrer SDK-Bibliothek verwendet. Verwenden Sie keine Bindestriche.

   c. Geben Sie für Java Build System **`maven`** oder **`gradle`** ein, um das Build-System anzugeben.

   d. Geben Sie für Java Group Id einen Gruppenbezeichner für Ihr SDK-Projekt ein. Geben Sie z. B. ei **`my-apig-api-examples`**. Dieser Bezeichner entspricht dem `<groupId>`-Tag unter `<project>` in der `pom.xml`-Datei, die sich im Projektordner des SDK befindet.

   e. Geben Sie für Java Artifact Id einen Artefaktbezeichner für Ihr SDK-Projekt ein. Geben Sie z. B. ei **`simple-calc-sdk`**. Dieser Bezeichner entspricht dem `<artifactId>`-Tag unter `<project>` in der `pom.xml`-Datei, die sich im Projektordner des SDK befindet.

   f. Geben Sie für Java Artifact Version eine Versions-ID-Zeichenfolge ein. Beispiel, **`1.0.0`**. Diese Version entspricht dem `<version>`-Tag unter `<project>` in der `pom.xml`-Datei, die sich im Projektordner des SDK befindet.

   g. Geben Sie für Source Code License Text den Lizenztext Ihres Quellcodes ein, falls vorhanden.

7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in Von API Gateway generiertes Java-SDK für eine REST-API verwenden zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des Android SDKs einer API

So generieren Sie das Android SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Android-Plattform aus und gehen Sie wie folgt vor:
   a. Geben Sie für Group ID den eindeutigen Bezeichner für das entsprechende Projekt ein. Dieser wird in der `pom.xml`-Datei verwendet (zum Beispiel **`com.mycompany`**).
   b. Geben Sie für Invoker package den Namespace für die generierten Client-Klassen ein (zum Beispiel **`com.mycompany.clientsdk`**).
   c. Geben Sie für Artifact ID den Namen der kompilierten JAR-Datei ohne die Version ein. Dieser wird in der `pom.xml`-Datei verwendet (zum Beispiel **`aws-apigateway-api-sdk`**).
   d. Geben Sie für Artifact version die Artefakt-Versionsnummer für den generierten Client ein. Dieser wird in der `pom.xml`-Datei verwendet und sollte dem Muster *`major.minor.patch`* folgen (zum Beispiel **`1.0.0`**).
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in Ein von API Gateway generiertes Android-SDK für eine REST-API verwenden zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des iOS SDKs einer API

So generieren Sie das iOS SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie für Plattform die Plattform iOS (Objective-C) oder iOS (Swift) aus und gehen Sie wie folgt vor:

   - Geben Sie für im Feld Prefix ein eindeutiges Präfix ein.

     Das Präfix hat folgende Auswirkungen: Wenn Sie beispielsweise als Präfix für das SDK der SimpleCalcAPI `result` Modelle mit, **`SIMPLE_CALC`** und zuweise`input`, `output` enthält das generierte SDK die `SIMPLE_CALCSimpleCalcClient` Klasse, die die API kapselt, einschließlich der Methodenanforderungen/-antworten. Darüber hinaus enthält das generierte SDK die `SIMPLE_CALCinput`-, `SIMPLE_CALCoutput`- und `SIMPLE_CALCresult`-Klassen, um Eingabe, Ausgabe und Ergebnisse bzw. die Anforderungseingabe und Antwortausgabe zu repräsentieren. Weitere Informationen finden Sie unter Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden.
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in Von API Gateway generiertes iOS-SDK für eine REST-API in Objective-C oder Swift verwenden zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren Sie das JavaScript SDK einer REST-API

Um das JavaScript SDK einer API in API Gateway zu generieren

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.

5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die JavaScriptPlattform aus.
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in Verwenden Sie ein von API Gateway generiertes JavaScript SDK für eine REST-API zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren des Ruby SDK einer API

So generieren Sie das Ruby SDK einer API in API Gateway

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie Stages.
4. Wählen Sie im Bereich Stufen den Namen der Stufe aus.
5. Öffnen Sie das Menü Stufenaktionen und wählen Sie dann SDK generieren aus.
6. Wählen Sie unter Plattform die Ruby-Plattform aus und gehen Sie wie folgt vor:

    a. Geben Sie für Service Name den Namen Ihres SDK an. Beispiel, **`SimpleCalc`**. Dieser Name wird verwendet, um den Ruby Gem Namespace Ihrer API zu generieren. Der Name besteht nur aus Buchstaben (`a-zA-Z`) ohne Sonderzeichen oder Zahlen.

    b. Geben Sie im Feld Ruby Gem Name den Namen des Ruby Gem ein, das den generierten SDK-Quellcode für Ihre API enthalten soll. Standardmäßig ist dies der Servicename in Kleinbuchstaben plus `-sdk`-Suffix, z. B. **`simplecalc-sdk`**.

    c. Geben Sie für Ruby Gem Version eine Versionsnummer für das generierte Ruby Gem ein. Standardmäßig ist der Wert eingestell `1.0.0`.
7. Wählen Sie Generate SDK (SDK erzeugen) aus und folgen Sie dann den Anweisungen auf dem Bildschirm, um das von API Gateway generierte SDK herunterzuladen.

Befolgen Sie die Anweisungen in Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden zur Verwendung des generierten SDKs.

Bei jeder Aktualisierung einer API müssen Sie die API erneut bereitstellen und das SDK neu generieren, damit die Aktualisierungen berücksichtigt werden.

Generieren Sie SDKs für eine API mithilfe von Befehlen AWS CLI

Sie können AWS CLI es verwenden, um ein SDK einer API für eine unterstützte Plattform zu generieren und herunterzuladen, indem Sie den Befehl get-sdk aufrufen. Im folgenden Abschnitt wird dieser Vorgang für einige der unterstützten Plattformen beschrieben.

Themen

Generieren Sie das SDK für Java für Android und laden Sie es herunter, indem Sie AWS CLI

Zum Erstellen und Herunterladen eines von API Gateway generierten Java für Android SDKs einer API (`udpuvvzbkc`) auf einer bestimmten Stufe (`test`) rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type android \
            --parameters groupId='com.mycompany',\
                invokerPackage='com.mycompany.myApiSdk',\
                artifactId='myApiSdk',\
                artifactVersion='0.0.1' \
            ~/apps/myApi/myApi-android-sdk.zip
```

Die letzte Eingabe von `~/apps/myApi/myApi-android-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-android-sdk.zip`.

Generieren Sie das JavaScript SDK und laden Sie es herunter mit dem AWS CLI

Um ein vom API Gateway generiertes JavaScript SDK einer API () zu einem bestimmten Zeitpunkt (`udpuvvzbkctest`) zu generieren und herunterzuladen, rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
```

```
            --sdk-type javascript \
            ~/apps/myApi/myApi-js-sdk.zip
```

Die letzte Eingabe von `~/apps/myApi/myApi-js-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-js-sdk.zip`.

Generieren Sie das Ruby-SDK und laden Sie es herunter mit dem AWS CLI

Zum Erstellen und Herunterladen eines Ruby SDKs einer API (`udpuvvzbkc`) auf einer bestimmten Stufe (`test`) rufen Sie den Befehl wie folgt auf:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test  \
            --sdk-type ruby \
            --parameters service.name=myApiRubySdk,ruby.gem-name=myApi,ruby.gem-
version=0.01 \
            ~/apps/myApi/myApi-ruby-sdk.zip
```

Die letzte Eingabe von `~/apps/myApi/myApi-ruby-sdk.zip` ist der Pfad zur heruntergeladenen SDK-Datei mit dem Namen `myApi-ruby-sdk.zip`.

Als Nächstes zeigen wir, wie Sie mit dem generierten SDK die zugrunde liegende API aufrufen. Weitere Informationen finden Sie unter REST-API im Amazon API Gateway aufrufen.

Verkaufen Sie Ihre API Gateway Gateway-APIs über AWS Marketplace

Nachdem Sie Ihre APIs erstellt, getestet und bereitgestellt haben, können Sie sie in einem API Gateway Gateway-Nutzungsplan verpacken und den Plan als SaaS-Produkt (Software as a Service) verkaufen AWS Marketplace. API-Käufern, die Ihr Produktangebot abonnieren, wird auf der AWS Marketplace Grundlage der Anzahl der Anfragen an den Nutzungsplan abgerechnet.

Um Ihre APIs weiterverkaufen zu können AWS Marketplace, müssen Sie den Vertriebskanal für die Integration AWS Marketplace mit API Gateway einrichten. Im Allgemeinen beinhaltet dies, dass Sie Ihr Produkt anbieten AWS Marketplace, eine IAM-Rolle mit entsprechenden Richtlinien einrichten, damit API Gateway Nutzungsdaten an diese senden kann AWS Marketplace, ein AWS Marketplace Produkt einem API Gateway Gateway-Nutzungsplan zuordnen und einem AWS Marketplace Käufer einen API-Gateway-API-Schlüssel zuordnen. Details dazu finden Sie in den folgenden Abschnitten.

Weitere Informationen zum Verkauf Ihrer API als SaaS-Produkt finden Sie im AWS Marketplace Benutzerhandbuch. AWS Marketplace

Themen

Initialisierung der AWS Marketplace -Integration mit API Gateway

Die folgenden Aufgaben betreffen die einmalige Initialisierung der AWS Marketplace Integration mit API Gateway, sodass Sie Ihre APIs als SaaS-Produkt verkaufen können.

Bieten Sie ein Produkt auf AWS Marketplace

Übermitteln Sie für die Eintragung Ihres Nutzungsplans als SaaS-Produkt ein Produktladeformular über AWS Marketplace. Das Produkt muss eine Dimension mit dem Namen `apigateway` vom Typ `requests` enthalten. Diese Dimension definiert die price-per-request und wird von API Gateway verwendet, um Anfragen an Ihre APIs zu messen.

Erstellen der Messungsrolle

Erstellen Sie eine IAM-Rolle mit dem Namen `ApiGatewayMarketplaceMeteringRole` und der folgenden Ausführungsrichtlinie und Vertrauensrichtlinie. Diese Rolle ermöglicht es API Gateway, in Ihrem Namen Nutzungsmetriken AWS Marketplace an zu senden.

Ausführungsrichtlinie der Messungsrolle

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Vertrauensbeziehungsrichtlinie der Messungsrolle

```
{
```

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

Verknüpfen Sie den Nutzungsplan mit dem AWS Marketplace Produkt

Wenn Sie ein Produkt anbieten AWS Marketplace, erhalten Sie einen AWS Marketplace Produktcode. Um API Gateway zu integrieren AWS Marketplace, verknüpfen Sie Ihren Nutzungsplan mit dem AWS Marketplace Produktcode. Sie aktivieren die Zuordnung, indem Sie das `productCode`Feld `UsagePlan` des API-Gateways auf Ihren AWS Marketplace Produktcode setzen, indem Sie die API Gateway-Konsole, die API Gateway Gateway-REST-API, die AWS CLI für API Gateway oder das AWS SDK für API Gateway verwenden. Das folgende Codebeispiel verwendet die API Gateway-REST-API:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/productCode",
        "value" : "MARKETPLACE_PRODUCT_CODE",
        "op" : "replace"
    }]
}
```

Verarbeiten von Kundenabonnements eines Nutzungsplans

Die folgenden Aufgaben werden von Ihrer Entwicklerportalanwendung übernommen.

Wenn ein Kunde Ihr Produkt abonniert AWS Marketplace, AWS Marketplace leitet eine `POST` Anfrage an die SaaS-Abonnement-URL weiter, unter der Sie Ihr Produkt angeboten haben. AWS Marketplace Die `POST`-Anforderung enthält einen `x-amzn-marketplace-token`-Parameter mit

Käuferinformationen. Befolgen Sie die Anweisungen unter Onboarding für SaaS-Kunden, um diese Umleitung in Ihrer Entwicklerportalanwendung zu verarbeiten.

Wenn Sie auf die Abonnementanfrage eines Kunden antworten, wird eine `subscribe-success` Benachrichtigung zu einem Amazon SNS SNS-Thema AWS Marketplace gesendet, das Sie abonnieren können. (Informationen finden Sie unter Onboarding für SaaS-Kunden). Um die Abonnementanfrage des Kunden anzunehmen, bearbeiten Sie die `subscribe-success` Benachrichtigung, indem Sie einen AWS Marketplace API-Gateway-API-Schlüssel für den Kunden erstellen oder abrufen, den vom Kunden bereitgestellten `customerId` API-Schlüsseln zuordnen und den API-Schlüssel dann zu Ihrem Nutzungsplan hinzufügen.

Wenn die Abonnementanforderung des Kunden abgeschlossen ist, sollte der Kunde in der Entwicklerportalanwendung mit dem zugeordneten API-Schlüssel angezeigt werden und wird informiert, dass der API-Schlüssel im Header `x-api-key` von Anforderungen an die APIs enthalten sein muss.

Wenn ein Kunde ein Abonnement für einen Nutzungsplan kündigt, AWS Marketplace sendet er eine `unsubscribe-success` Benachrichtigung zum Thema SNS. Zum Abschließen der Abonnementkündigung des Kunden verarbeiten Sie die `unsubscribe-success`-Benachrichtigung, indem Sie die API-Schlüssel des Kunden aus dem Nutzungsplan entfernen.

Autorisieren eines Kunden für den Zugriff auf den Nutzungsplan

Verwenden Sie zum Autorisieren des Zugriffs eines Kunden auf Ihren Nutzungsplan die API Gateway-API, um einen API-Schlüssel abzurufen oder zu erstellen, und fügen Sie dem Nutzungsplan den API-Schlüssel hinzu.

Das folgende Beispiel zeigt, wie die API Gateway REST API aufgerufen wird, um einen neuen API-Schlüssel mit einem bestimmten AWS Marketplace `customerId` Wert (*`MARKETPLACE_CUSTOMER_ID`*) zu erstellen.

```
POST apikeys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "name" : "my_api_key",
  "description" : "My API key",
  "enabled" : "false",
  "stageKeys" : [ {
    "restApiId" : "uycll6xg9a",
```

```
    "stageName" : "prod"
  } ],
  "customerId" : "MARKETPLACE_CUSTOMER_ID"
}
```

Das folgende Beispiel zeigt, wie Sie einen API-Schlüssel mit einem bestimmten AWS Marketplace customerId Wert (MARKETPLACE_CUSTOMER_ID) abrufen.

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Erstellen Sie zum Hinzufügen eines API-Schlüssels zu einem Nutzungsplan einen UsagePlanKey mit dem API-Schlüssel für den entsprechenden Nutzungsplan. Im folgenden Beispiel wird gezeigt, wie dies über die API Gateway-REST-API durchgeführt wird, wobei `n371pt` die Nutzungsplan-ID und `q5ugs7qjjh` eine Beispiel-API-`keyId` ist, die in den vorhergehenden Beispielen zurückgegeben wurde.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "keyId": "q5ugs7qjjh",
    "keyType": "API_KEY"
}
```

Verknüpfen eines Kunden mit einem API-Schlüssel

Sie müssen das `customerId` Feld mit der ApiKeyKunden-ID des AWS Marketplace Kunden aktualisieren. Auf diese Weise wird der API-Schlüssel mit dem AWS Marketplace -Kunden verknüpft, was die Messung und Abrechnung dieses Käufers ermöglicht. Im folgenden Codebeispiel wird die API Gateway-REST-API aufgerufen, um dies durchzuführen.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
```

```
        "path" : "/customerId",
        "value" : "MARKETPLACE_CUSTOMER_ID",
        "op" : "replace"
    }]
}
```

REST-API schützen

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, wie böswillige Benutzer oder Spitzen im Datenverkehr. Sie können Ihre API mit Strategien schützen, wie das Generieren von SSL-Zertifikaten, das Konfigurieren einer Firewall für Webanwendungen, das Festlegen von Drosselungs-Targets und das Ermöglichen des Zugriffs auf Ihre API über eine Virtual Private Cloud (VPC). In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen mit API Gateway aktivieren können.

Themen

- Konfigurieren der gegenseitigen TLS-Authentifizierung für eine REST-API
- Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-Authentifizierung
- AWS WAF Zum Schutz Ihrer APIs verwenden
- Drosseln der API-Anforderungen für einen besseren Durchsatz
- Private REST-APIs im Amazon API Gateway

Konfigurieren der gegenseitigen TLS-Authentifizierung für eine REST-API

Die gegenseitige TLS-Authentifizierung erfordert eine bidirektionale Authentifizierung zwischen dem Client und dem Server. Bei gegenseitiger TLS müssen Clients X.509-Zertifikate zur Prüfung ihrer Identität präsentieren, um auf Ihre API zuzugreifen. Gegenseitiges TLS ist eine allgemeine Anforderung für das Internet der Dinge (IoT) und business-to-business Anwendungen.

Sie können gegenseitiges TLS zusammen mit anderen Autorisierungs- und Authentifizierungsvorgängen verwenden, die API Gateway unterstützt. API Gateway leitet die von Kunden bereitgestellten Zertifikate an Lambda-Genehmiger und an Backend-Integrationen weiter.

> **Important**
>
> Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Clients nur mithilfe eines

benutzerdefinierten Domänennamens mit gegenseitiger TLS auf Ihre API zugreifen können, deaktivieren Sie den `execute-api`-Standardendpunkt. Weitere Informationen hierzu finden Sie unter the section called “Deaktivieren des Standardendpunkts”.

Themen

Voraussetzungen für gegenseitige TLS

Um gegenseitige TLS zu konfigurieren, benötigen Sie:

- Einen benutzerdefinierten Domänennamen
- AWS Certificate Manager Für Ihren benutzerdefinierten Domainnamen ist mindestens ein Zertifikat konfiguriert
- Ein in Amazon S3 konfigurierter und hochgeladener Truststore

Benutzerdefinierte Domänennamen

Um die gegenseitige TLS-Authentifizierung für eine REST-API zu aktivieren, müssen Sie einen benutzerdefinierten Domänennamen für Ihre API konfigurieren. Sie können gegenseitige TLS für einen benutzerdefinierten Domänennamen aktivieren und dann den benutzerdefinierten Domänennamen für Clients bereitstellen. Um mithilfe eines benutzerdefinierten Domänennamens auf eine API zuzugreifen, für die gegenseitige TLS aktiviert ist, müssen Clients Zertifikate präsentieren, denen Sie in API-Anforderungen vertrauen. Sie finden mehr Informationen in the section called “Benutzerdefinierte Domänennamen”.

Verwendung AWS Certificate Manager ausgestellter Zertifikate

Sie können ein öffentlich vertrauenswürdiges Zertifikat direkt von ACM anfordern oder öffentliche oder selbstsignierte Zertifikate importieren. Gehen Sie zum Einrichten eines Zertifikats in ACM zu ACM. Wenn Sie ein Zertifikat importieren möchten, lesen Sie weiter im folgenden Abschnitt.

Verwenden eines importierten AWS Private Certificate Authority Zertifikats oder Zertifikats

Um ein in ACM importiertes Zertifikat oder ein Zertifikat von Mutual TLS verwenden AWS Private Certificate Authority zu können, benötigt API Gateway ein von ACM `ownershipVerificationCertificate` ausgestelltes. Dieses Eigentumszertifikat wird nur verwendet, um zu überprüfen, ob Sie über Berechtigungen zur Verwendung des Domänennamens verfügen. Es wird nicht für den TLS-Handshake verwendet. Wenn Sie noch keine `ownershipVerificationCertificate` haben, navigieren Sie zuhttps://console.aws.amazon.com/acm/, um eine einzurichten.

Sie müssen dieses Zertifikat für die gesamte Lebensdauer Ihres Domainnamens gültig halten. Wenn ein Zertifikat abläuft und die automatische Verlängerung fehlschlägt, werden alle Updates des Domänennamens gesperrt. Sie müssen die `ownershipVerificationCertificateArn` mit einem gültigen `ownershipVerificationCertificate` aktualisieren, bevor Sie weitere Änderungen vornehmen können. Die `ownershipVerificationCertificate` kann nicht als Serverzertifikat für eine andere gemeinsame TLS-Domäne in API Gateway verwendet werden. Wenn ein Zertifikat direkt wieder in ACM importiert wird, muss der Aussteller gleich bleiben.

Konfigurieren Ihres Truststore

Truststores sind Textdateien mit einer `.pem`-Dateierweiterung. Sie sind eine vertrauenswürdige Liste von Zertifikaten von Zertifizierungsstellen. Um gegenseitige TLS zu verwenden, erstellen Sie einen Truststore von X.509-Zertifikaten, denen Sie vertrauen, für den Zugriff auf Ihre API.

Sie müssen die vollständige Vertrauenskette vom ausstellenden CA-Zertifikat bis zum Stamm-CA-Zertifikat in Ihren Truststore aufnehmen. API Gateway akzeptiert Clientzertifikate, die von jeder Zertifizierungsstelle in der Vertrauenskette ausgestellt werden. Die Zertifikate können von öffentlichen oder privaten Zertifizierungsstellen stammen. Zertifikate können eine maximale Kettenlänge von vier haben. Sie können auch selbstsignierte Zertifikate bereitstellen. Die folgenden Algorithmen werden im Truststore unterstützt:

- SHA-256 oder stärker
- RSA-2048 oder stärker
- ECDSA-256 oder ECDSA-384

API Gateway validiert eine Reihe von Zertifikatseigenschaften. Sie können Lambda-Genehmiger verwenden, um zusätzliche Prüfungen durchzuführen, wenn ein Client eine API aufruft, einschließlich der Prüfung, ob ein Zertifikat widerrufen wurde. API Gateway validiert die folgenden Eigenschaften:

Validierung	Beschreibung
X.509-Syntax	Das Zertifikat muss die X.509-Syntaxanford erungen erfüllen.
Integrität	Der Inhalt des Zertifikats darf sich nicht von dem unterscheiden, was von der Zertifizi erungsstelle des Truststore signiert wurde.
Gültigkeit	Die Gültigkeitsdauer des Zertifikats muss aktuell sein.
Namensverkettung/Schlüsselverkettung	Die Verkettung der Namen und Antragssteller von Zertifikaten darf nicht unterbrochen sein. Zertifikate können eine maximale Kettenlänge von vier haben.

Hochladen des Vertrauensspeichers in einen Amazon-S3-Bucket in einer einzigen Datei.

Im Folgenden finden Sie ein Beispiel dafür, wie eine .pem-Datei aussehen könnte.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Der folgende AWS CLI Befehl lädt `certificates.pem` in Ihren Amazon S3 S3-Bucket hoch.

```
aws s3 cp certificates.pem s3://bucket-name
```

Ihr Amazon S3 S3-Bucket muss über Leseberechtigungen für API Gateway verfügen, damit API Gateway auf Ihren Truststore zugreifen kann.

Konfigurieren von gegenseitiger TLS für einen benutzerdefinierten Domänennamen

Um Mutual TLS für eine REST-API zu konfigurieren, müssen Sie einen regionalen benutzerdefinierten Domainnamen für Ihre API mit einer TLS_1_2 Sicherheitsrichtlinie verwenden. Weitere Informationen zur Auswahl einer Sicherheitsrichtlinie finden Sie unterthe section called “Auswahl einer Sicherheitsrichtlinie”.

> **Note**
>
> Gegenseitiges TLS wird für private APIs nicht unterstützt.

Nachdem Sie Ihren Vertrauensspeicher in Amazon S3 hochgeladen haben, können Sie Ihren benutzerdefinierten Domänennamen für die Verwendung von gegenseitigem TLS konfigurieren. Fügen Sie Folgendes (Schrägstriche enthalten) in ein Terminal ein:

```
aws apigateway create-domain-name --region us-east-2 \
    --domain-name api.example.com \
    --regional-certificate-arn arn:aws:acm:us-
east-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --endpoint-configuration types=REGIONAL \
    --security-policy TLS_1_2 \
    --mutual-tls-authentication truststoreUri=s3://bucket-name/key-name
```

Nachdem Sie den Domänennamen erstellt haben, müssen Sie DNS-Datensätze und Basepath-Mappings für API-Vorgänge konfigurieren. Weitere Informationen hierzu finden Sie unter Einrichten eines regionalen benutzerdefinierten Domänennamens in API Gateway.

Aufrufen einer API mithilfe eines benutzerdefinierten Domänennamens, der gegenseitige TLS erfordert

Um eine API mit aktivierter gegenseitiger TLS aufzurufen, müssen Clients in der API-Anforderung ein vertrauenswürdiges Zertifikat präsentieren. Wenn ein Client versucht, Ihre API aufzurufen, sucht API Gateway in Ihrem Truststore nach dem Aussteller des Clientzertifikats. Damit API Gateway mit

der Anforderung fortfahren kann, müssen sich der Aussteller des Zertifikats und die vollständige Vertrauenskette bis zum Stamm-CA-Zertifikats in Ihrem Truststore befinden.

Mit dem folgenden `curl`-Beispielbefehl wird eine Anforderung an `api.example.com`, gesendet, die `my-cert.pem` in die Anforderung einschließt. `my-key.key` ist der private Schlüssel für das Zertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Ihre API wird nur aufgerufen, wenn Ihr Truststore dem Zertifikat vertraut. Die folgenden Bedingungen führen dazu, dass API Gateway den TLS-Handshake fehlschlägt und die Anforderung mit einem `403`-Statuscode verweigert. Wenn Ihr Zertifikat:

- nicht vertrauenswürdig ist
- abgelaufen ist
- keinen unterstützten Algorithmus verwendet

> **Note**
>
> API Gateway überprüft nicht, ob ein Zertifikat widerrufen wurde.

Aktualisieren Ihres Truststore

Um die Zertifikate in Ihrem Truststore zu aktualisieren, laden Sie ein neues Zertifikatspaket in Amazon S3 hoch. Anschließend können Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um das aktualisierte Zertifikat zu verwenden.

Verwenden Sie Amazon-S3-Versioning, um mehrere Versionen Ihres Vertrauensspeichers zu verwalten. Wenn Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um eine neue Vertrauensspeicher-Version zu verwenden, gibt API Gateway Warnungen zurück, wenn Zertifikate ungültig sind.

API Gateway erzeugt Zertifikatswarnungen nur dann, wenn Sie Ihren Domänennamen aktualisieren. API Gateway benachrichtigt Sie nicht, wenn ein zuvor hochgeladenes Zertifikat abläuft.

Mit dem folgenden AWS CLI Befehl wird ein benutzerdefinierter Domainname aktualisiert, sodass er eine neue Truststore-Version verwendet.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

Gegenseitige TLS deaktivieren

Um gegenseitige TLS für einen benutzerdefinierten Domänennamen zu deaktivieren, entfernen Sie den Truststore aus Ihrem benutzerdefinierten Domänennamen, wie im folgenden Befehl gezeigt.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

Fehlerbehebung bei Zertifikatwarnungen

Wenn Sie einen benutzerdefinierten Domänennamen mit gegenseitiger TLS erstellen, gibt API Gateway Warnungen zurück, wenn Zertifikate im Truststore ungültig sind. Dies kann auch auftreten, wenn ein benutzerdefinierter Domänenname aktualisiert wird, um einen neuen Truststore zu verwenden. Die Warnungen geben das Problem mit dem Zertifikat und den Betreff des Zertifikats an, das die Warnung erstellt hat. Gegenseitige TLS ist weiterhin für Ihre API aktiviert, aber einige Clients können möglicherweise nicht auf Ihre API zugreifen.

Um aufzudecken, welches Zertifikat die Warnung erzeugt hat, dekodieren Sie die Zertifikate in Ihrem Truststore. Sie können Tools wie `openssl` zum Dekodieren der Zertifikate und zur Identifizierung ihrer Subjekte verwenden.

Der folgende Befehl zeigt den Inhalt eines Zertifikats einschließlich seines Betreffs an:

```
openssl x509 -in certificate.crt -text -noout
```

Aktualisieren oder entfernen Sie die Zertifikate, die Warnungen erstellt haben, und laden Sie dann einen neuen Truststore in Amazon S3 hoch. Nachdem Sie einen neuen Truststore hochgeladen haben, aktualisieren Sie Ihren benutzerdefinierten Domänennamen, um den neuen Truststore zu verwenden.

Problembehandlung bei Domänennamenskonflikten

Der Fehler `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` bedeutet, dass mehrere Zertifizierungsstellen ein Zertifikat für diese Domäne ausgestellt haben. Für jeden Subjekt im Zertifikat kann es nur einen Aussteller in API Gateway für gegenseitige TLS-Domänen geben. Sie müssen alle Ihre Zertifikate für dieses Thema über einen einzigen Aussteller abrufen. Wenn das Problem mit einem Zertifikat besteht, das Sie nicht kontrollieren können, Sie jedoch den Besitz des Domänennamens nachweisen können, kontaktieren Sie AWS Support, um ein Ticket zu eröffnen.

Fehlerbehebung bei Statusmeldungen für Domänennamen

`PENDING_CERTIFICATE_REIMPORT`: Das bedeutet, dass Sie ein Zertifikat erneut in ACM importiert haben und die Validierung fehlgeschlagen ist, da das neue Zertifikat über ein SAN (alternativer Antragstellername) verfügt, das nicht von der `ownershipVerificationCertificate` abgedeckt ist oder der Betreff oder die SANs im Zertifikat decken den Domänennamen nicht ab. Möglicherweise ist etwas falsch konfiguriert oder ein ungültiges Zertifikat wurde importiert. Sie müssen ein gültiges Zertifikat erneut in ACM importieren. Weitere Informationen zur Validierung finden Sie unter Validierung des Domänenbesitzes.

`PENDING_OWNERSHIP_VERIFICATION`: Dies bedeutet, dass Ihr zuvor verifiziertes Zertifikat abgelaufen ist und ACM es nicht automatisch erneuern konnte. Sie müssen das Zertifikat erneuern oder ein neues Zertifikat anfordern. Weitere Informationen zur Zertifikatserneuerung finden Sie im Leitfaden ACMs Fehlerbehebung bei der Erneuerung verwalteter Zertifikate.

Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-Authentifizierung

Sie können API Gateway verwenden, um ein SSL-Zertifikat zu generieren und dann dessen öffentlichen Schlüssel im Backend verwendet, um zu überprüfen, ob HTTP-API-Anfragen an Ihr Backend-System von API Gateway stammen. So kann Ihr HTTP-Backend nur Anfragen steuern und akzeptieren, die von Amazon API Gateway stammen – selbst wenn das Backend öffentlich zugänglich ist.

> **Note**
>
> Einige Backend-Server unterstützen möglicherweise keine SSL-Client-Authentifizierung als API Gateway. Sie könnten einen SSL-Zertifikatsfehler zurückgeben. Eine Liste der inkompatiblen Backend-Server finden Sie unter the section called “Wichtige Hinweise”.

Die SSL-Zertifikate, die von API Gateway generiert werden, sind selbstsigniert, und nur der öffentliche Schlüssel eines Zertifikats ist in der API Gateway-Konsole oder über die APIs sichtbar.

Themen

Client-Zertifikat über die API Gateway-Konsole generieren

1. Öffnen Sie die API Gateway-Konsole unter https://console.aws.amazon.com/apigateway/.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsfenster Client Certificates (Clientzertifikate).
4. Wählen Sie auf der Seite Client Certificates (Clientzertifikate) die Option Generate certificate (Zertifikat generieren) aus.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
6. Wählen Sie Generate certificate (Zertifikat generieren) aus, um das Zertifikat zu generieren. API Gateway generiert ein neues Zertifikat und gibt die neue Zertifikats-GUID zusammen mit dem PEM-kodierten öffentlichen Schlüssel zurück.

Sie können eine API nun für die Verwendung des Zertifikats konfigurieren.

Konfigurieren einer API für die Verwendung von SSL-Zertifikaten

Bei diesen Anleitungen wird davon ausgegangen, dass Sie bereits abgeschlossen habe Client-Zertifikat über die API Gateway-Konsole generieren.

1. Erstellen oder öffnen Sie in der API Gateway-Konsole eine API, für die Sie das Client-Zertifikat verwenden möchten. Stellen Sie sicher, dass die API für eine Stufe bereitgestellt wurde.
2. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
3. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Client certificate (Clientzertifikat) ein Zertifikat aus.
5. Wählen Sie Änderungen speichern aus.

   Wenn die API zuvor in der API Gateway-Konsole bereitgestellt wurde, müssen Sie sie neu bereitstellen, damit die Änderungen wirksam werden. Weitere Informationen finden Sie unter the section called “Erneutes Bereitstellen einer REST-API für eine Stufe”.

Nachdem ein Zertifikat für die API ausgewählt und gespeichert wurde, verwendet API Gateway das Zertifikat für alle Aufrufe an HTTP-API-Integrationen in Ihrer API.

Testaufruf zur Überprüfung der Konfiguration des Clientzertifikats

1. Wählen Sie eine API-Methode. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte Test anzuzeigen.
2. Wählen Sie für Client certificate (Clientzertifikat) ein Zertifikat aus.
3. Wählen Sie Test aus.

API Gateway nutzt das ausgewählte SSL-Zertifikat für das HTTP-Backend, um die API zu authentifizieren.

Konfigurieren eines Backend-HTTPS-Servers zur Überprüfung des Clientzertifikats

Diese Anleitungen setzen voraus, dass Sie die in Client-Zertifikat über die API Gateway-Konsole generieren beschriebenen Aktionen ausgeführt und eine Kopie des Clientzertifikats heruntergeladen haben. Sie können ein Client-Zertifikat herunterladen, indem Sie `clientcertificate:by-id` der API-Gateway-REST-API oder `get-client-certificate` der AWS CLI aufrufen.

Vor der Konfiguration eines Backend-HTTPS-Servers zur Überprüfung des SSL-Clientzertifikats von Amazon API Gateway müssen Sie den PEM-kodierten privaten Schlüssel und ein von einer vertrauenswürdigen Zertifizierungsstelle bereitgestelltes serverseitiges Zertifikat erhalten haben.

Wenn der Name der Serverdomäne `myserver.mydomain.com` ist, muss der CNAME-Wert des Serverzertifikats `myserver.mydomain.com` oder `*.mydomain.com` sein.

Zu den unterstützten Zertifizierungsstellen gehören Let's Encrypt oder eine der in the section called "Unterstützte Zertifizierungsstellen für die HTTP- und HTTP-Proxy-Integration" aufgelisteten Stellen.

Angenommen, die Clientzertifikatdatei ist `apig-cert.pem` und der private Schlüssel und die Zertifikatdateien des Servers sind `server-key.pem` bzw. `server-cert.pem`. Für einen Node.js-Server im Backend können Sie den Server wie folgt konfigurieren:

```
var fs = require('fs');
var https = require('https');
var options = {
    key: fs.readFileSync('server-key.pem'),
    cert: fs.readFileSync('server-cert.pem'),
    ca: fs.readFileSync('apig-cert.pem'),
    requestCert: true,
    rejectUnauthorized: true
};
https.createServer(options, function (req, res) {
    res.writeHead(200);
    res.end("hello world\n");
}).listen(443);
```

Für eine Node-Express-App können Sie die client-certificate-authModule verwenden, um Client-Anfragen mit PEM-codierten Zertifikaten zu authentifizieren.

Informationen zu anderen HTTPS-Servers finden Sie in der Dokumentation zum Server.

Rotieren ablaufender Clientzertifikate

Das von API Gateway generierte Client-Zertifikat ist 365 Tage gültig. Sie müssen das Zertifikat rotieren, bevor ein Clientzertifikat auf einer API-Stufe abläuft, um Ausfallzeiten für die API zu vermeiden. Sie können das Ablaufdatum des Zertifikats überprüfen, indem Sie clientCertificate:by-ID der API Gateway Gateway-REST-API aufrufen oder den AWS CLI Befehl von aufrufen get-client-certificateund die zurückgegebene Eigenschaft expirationDate überprüfen.

Gehen Sie folgendermaßen vor, um ein Clientzertifikat zu rotieren:

1. Generieren Sie ein neues Client-Zertifikat, indem Sie clientcertificate:generate der API Gateway REST API oder den Befehl von aufrufen. AWS CLI generate-client-certificate In diesem Tutorial wird davon ausgegangen, dass die ID des neuen Clientzertifikats laute `ndiqef`.
2. Aktualisieren Sie den Backend-Server, sodass dieser das neue Clientzertifikat enthält. Entfernen Sie das vorhandene Clientzertifikat noch nicht.

   Einige Server erfordern möglicherweise einen Neustart, um die Aktualisierung abzuschließen. Informieren Sie sich in der Serverdokumentation, um festzustellen, ob Sie den Server während der Aktualisierung neu starten müssen.
3. Aktualisieren Sie die API-Stufe, um das neue Client-Zertifikat zu verwenden, indem Sie stage:update der API Gateway-REST-API mit der neuen Client-Zertifikat-ID (`ndiqef`) aufrufen:

   ```
   PATCH /restapis/{restapi-id}/stages/stage1 HTTP/1.1
   Content-Type: application/json
   Host: apigateway.us-east-1.amazonaws.com
   X-Amz-Date: 20170603T200400Z
   Authorization: AWS4-HMAC-SHA256 Credential=...

   {
     "patchOperations" : [
       {
           "op" : "replace",
           "path" : "/clientCertificateId",
           "value" : "ndiqef"
       }
     ]
   }
   ```

   oder indem Sie den CLI-Befehl update-stage aufrufen.
4. Aktualisieren Sie den Backend-Server, um das alte Zertifikat zu entfernen.
5. Löschen Sie das alte Zertifikat von API Gateway, indem Sie clientcertificate:delete der API Gateway Gateway-REST-API aufrufen und dabei die clientCertificateId (`a1b2c3`) des alten Zertifikats angeben:

   ```
   DELETE /clientcertificates/a1b2c3
   ```

   oder indem Sie den CLI-Befehl aufrufen von delete-client-certificate:

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Gehen Sie folgendermaßen vor, um ein Clientzertifikat in der Konsole für eine zuvor bereitgestellte API zu rotieren:

1. Wählen Sie im Hauptnavigationsfenster Client Certificates (Clientzertifikate).
2. Wählen Sie im Bereich Client Certificates ((Clientzertifikate) Generate certificate (Zertifikat generieren) aus.
3. Öffnen Sie die API, für die Sie das Client-Zertifikat verwenden möchten.
4. Wählen Sie unter der ausgewählten API Stages und wählen Sie dann eine Stufe aus.
5. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
6. Wählen Sie für Client certificate (Clientzertifikat) das neue Zertifikat aus.
7. Wählen Sie zum Speichern der Einstellungen Save changes (Änderungen speichern).

   Sie müssen die API erneut bereitstellen, damit die Änderungen wirksam werden. Weitere Informationen finden Sie unter the section called “Erneutes Bereitstellen einer REST-API für eine Stufe”.

Von API Gateway unterstützte Autorisierungen für HTTP- und HTTP-Proxy-Integrationen

Die folgende Liste zeigt die Autorisierungen, die von API Gateway für HTTP- und HTTP-Proxy-Integrationen unterstützt werden.

```
Alias name: accvraiz1
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: acraizfnmtrcm
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F
Alias name: actalis
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
```

```
Alias name: actalisauthenticationrootca
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: addtrustclass1ca
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: addtrustexternalca
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: addtrustqualifiedca
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: affirmtrustcommercial
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustcommercialca
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustnetworking
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustnetworkingca
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustpremium
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumca
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumecc
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
```

```
Alias name: affirmtrustpremiumeccca
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: amazon-ca-g4-acm1
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8
Alias name: amazon-ca-g4-acm2
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B
Alias name: amazon-ca-g4-acm3
  SHA1: 7A:DB:56:57:5F:D6:EE:67:85:0A:64:BB:1C:E9:E4:B0:9A:DB:9D:07
  SHA256:
 6B:EB:9D:20:2E:C2:00:70:BD:D2:5E:D3:C0:C8:33:2C:B4:78:07:C5:82:94:4E:7E:23:28:22:71:A4:8E:0E:C
Alias name: amazon-ca-g4-legacy
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5
Alias name: amazon-root-ca-ecc-384-1
  SHA1: F9:5E:4A:AB:9C:2D:57:61:63:3D:B2:57:B4:0F:24:9E:7B:E2:23:7D
  SHA256:
 C6:BD:E5:66:C2:72:2A:0E:96:E9:C1:2C:BF:38:92:D9:55:4D:29:03:57:30:72:40:7F:4E:70:17:3B:3C:9B:6
Alias name: amazon-root-ca-rsa-2k-1
  SHA1: 8A:9A:AC:27:FC:86:D4:50:23:AD:D5:63:F9:1E:AE:2C:AF:63:08:6C
  SHA256:
 0F:8F:33:83:FB:70:02:89:49:24:E1:AA:B0:D7:FB:5A:BF:98:DF:75:8E:0F:FE:61:86:92:BC:F0:75:35:CC:8
Alias name: amazon-root-ca-rsa-4k-1
  SHA1: EC:BD:09:61:F5:7A:B6:A8:76:BB:20:8F:14:05:ED:7E:70:ED:39:45
  SHA256:
 36:AE:AD:C2:6A:60:07:90:6B:83:A3:73:2D:D1:2B:D4:00:5E:C7:F2:76:11:99:A9:D4:DA:63:2F:59:B2:8B:C
Alias name: amazon1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazon2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazon3
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
```

```
Alias name: amazon4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amazonrootca1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazonrootca2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazonrootca3
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazonrootca4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amzninternalinfoseccag3
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6
Alias name: amzninternalrootca
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A
Alias name: aolrootca1
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: aolrootca2
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: atostrustedroot2011
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
```

```
Alias name: baltimorecodesigningca
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8
Alias name: baltimorecybertrustca
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: baltimorecybertrustroot
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: buypassclass2ca
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass2rootca
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass3ca
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: buypassclass3rootca
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: cadisigrootr2
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: camerfirmachambersca
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: camerfirmachamberscommerceca
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: camerfirmachambersignca
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
```

```
Alias name: certigna
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: certignarootca
  SHA1: 2D:0D:52:14:FF:9E:AD:99:24:01:74:20:47:6E:6C:85:27:27:F5:43
  SHA256:
 D4:8D:3D:23:EE:DB:50:A4:59:E5:51:97:60:1C:27:77:4B:9D:7B:18:C9:4D:5A:05:95:11:A1:02:50:B9:31:6
Alias name: certplusclass2primaryca
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: certplusclass3pprimaryca
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8
Alias name: certsignrootca
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: certsignrootcag2
  SHA1: 26:F9:93:B4:ED:3D:28:27:B0:B9:4B:A7:E9:15:1D:A3:8D:92:E5:32
  SHA256:
 65:7C:FE:2F:A7:3F:AA:38:46:25:71:F3:32:A2:36:3A:46:FC:E7:02:09:51:71:07:02:CD:FB:B6:EE:DA:33:0
Alias name: certum2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: certumca
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: certumtrustednetworkca
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: certumtrustednetworkca2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: cfcaevroot
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F
```

```
Alias name: chambersofcommerceroot2008
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: chunghwaepkirootca
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: cia-crt-g3-01-ca
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E
Alias name: cia-crt-g3-02-ca
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C
Alias name: comodo-ca
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: comodoaaaca
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodoaaaservicesroot
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodocertificationauthority
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: comodoecccertificationauthority
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: comodorsacertificationauthority
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: cybertrustglobalroot
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
```

```
Alias name: deprecateditsecca
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C
Alias name: deutschetelekomrootca2
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: digicertassuredidrootca
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: digicertassuredidrootg2
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8
Alias name: digicertassuredidrootg3
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C
Alias name: digicertglobalrootca
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: digicertglobalrootg2
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5
Alias name: digicertglobalrootg3
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D
Alias name: digicerthighassuranceevrootca
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: digicerttrustedrootg4
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8
Alias name: dstrootcax3
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
```

```
Alias name: dtrustrootclass3ca22009
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: dtrustrootclass3ca2ev2009
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: ecacc
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9
Alias name: emsigneccrootcac3
  SHA1: B6:AF:43:C2:9B:81:53:7D:F6:EF:6B:C3:1F:1F:60:15:0C:EE:48:66
  SHA256:
 BC:4D:80:9B:15:18:9D:78:DB:3E:1D:8C:F4:F9:72:6A:79:5D:A1:64:3C:A5:F1:35:8E:1D:DB:0E:DC:0D:7E:B
Alias name: emsigneccrootcag3
  SHA1: 30:43:FA:4F:F2:57:DC:A0:C3:80:EE:2E:58:EA:78:B2:3F:E6:BB:C1
  SHA256:
 86:A1:EC:BA:08:9C:4A:8D:3B:BE:27:34:C6:12:BA:34:1D:81:3E:04:3C:F9:E8:A8:62:CD:5C:57:A3:6B:BE:6
Alias name: emsignrootcac1
  SHA1: E7:2E:F1:DF:FC:B2:09:28:CF:5D:D4:D5:67:37:B1:51:CB:86:4F:01
  SHA256:
 12:56:09:AA:30:1D:A0:A2:49:B9:7A:82:39:CB:6A:34:21:6F:44:DC:AC:9F:39:54:B1:42:92:F2:E8:C8:60:8
Alias name: emsignrootcag1
  SHA1: 8A:C7:AD:8F:73:AC:4E:C1:B5:75:4D:A5:40:F4:FC:CF:7C:B5:8E:8C
  SHA256:
 40:F6:AF:03:46:A9:9A:A1:CD:1D:55:5A:4E:9C:CE:62:C7:F9:63:46:03:EE:40:66:15:83:3D:C8:C8:D0:03:6
Alias name: entrust2048ca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustevca
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustnetpremium2048secureserverca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustrootcag2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
```

```
Alias name: entrustrootcertificationauthority
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustrootcertificationauthorityec1
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F
Alias name: entrustrootcertificationauthorityg2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthorityg4
  SHA1: 14:88:4E:86:26:37:B0:26:AF:59:62:5C:40:77:EC:35:29:BA:96:01
  SHA256:
 DB:35:17:D1:F6:73:2A:2D:5A:B9:7C:53:3E:C7:07:79:EE:32:70:A6:2F:B4:AC:42:38:37:24:60:E6:F0:1E:8
Alias name: epkirootcertificationauthority
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: equifaxsecureebusinessca1
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9
Alias name: equifaxsecureglobalebusinessca1
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A
Alias name: eszignorootca2017
  SHA1: 89:D4:83:03:4F:9E:9A:48:80:5F:72:37:D4:A9:A6:EF:CB:7C:1F:D1
  SHA256:
 BE:B0:0B:30:83:9B:9B:C3:2C:32:E4:44:79:05:95:06:41:F2:64:21:B1:5E:D0:89:19:8B:51:8A:E2:EA:1B:9
Alias name: etugracertificationauthority
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: gd-class2-root.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: gd_bundle-g2.pem
  SHA1: 27:AC:93:69:FA:F2:52:07:BB:26:27:CE:FA:CC:BE:4E:F9:C3:19:B8
  SHA256:
 97:3A:41:27:6F:FD:01:E0:27:A2:AA:D4:9E:34:C3:78:46:D3:E9:76:FF:6A:62:0B:67:12:E3:38:32:04:1A:A
```

```
Alias name: gdcatrustauthr5root
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9
Alias name: gdroot-g2.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: geotrustglobalca
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: geotrustprimaryca
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycag2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycag3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustprimarycertificationauthority
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycertificationauthorityg2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycertificationauthorityg3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustuniversalca
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: geotrustuniversalca2
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
```

```
Alias name: globalchambersignroot2008
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: globalsignca
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsigneccrootcar4
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8
Alias name: globalsigneccrootcar5
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2
Alias name: globalsignr2ca
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignr3ca
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootca
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsignrootcar2
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignrootcar3
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootcar6
  SHA1: 80:94:64:0E:B5:A7:A1:CA:11:9C:1F:DD:D5:9F:81:02:63:A7:FB:D1
  SHA256:
 2C:AB:EA:FE:37:D0:6C:A2:2A:BA:73:91:C0:03:3D:25:98:29:52:C4:53:64:73:49:76:3A:3A:B5:AD:6C:CF:6
Alias name: godaddyclass2ca
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
```

```
Alias name: godaddyrootcertificateauthorityg2
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: godaddyrootg2ca
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: gtsrootr1
  SHA1: E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
  SHA256:
 2A:57:54:71:E3:13:40:BC:21:58:1C:BD:2C:F1:3E:15:84:63:20:3E:CE:94:BC:F9:D3:CC:19:6B:F0:9A:54:7
Alias name: gtsrootr2
  SHA1: D2:73:96:2A:2A:5E:39:9F:73:3F:E1:C7:1E:64:3F:03:38:34:FC:4D
  SHA256:
 C4:5D:7B:B0:8E:6D:67:E6:2E:42:35:11:0B:56:4E:5F:78:FD:92:EF:05:8C:84:0A:EA:4E:64:55:D7:58:5C:6
Alias name: gtsrootr3
  SHA1: 30:D4:24:6F:07:FF:DB:91:89:8A:0B:E9:49:66:11:EB:8C:5E:46:E5
  SHA256:
 15:D5:B8:77:46:19:EA:7D:54:CE:1C:A6:D0:B0:C4:03:E0:37:A9:17:F1:31:E8:A0:4E:1E:6B:7A:71:BA:BC:E
Alias name: gtsrootr4
  SHA1: 2A:1D:60:27:D9:4A:B1:0A:1C:4D:91:5C:CD:33:A0:CB:3E:2D:54:CB
  SHA256:
 71:CC:A5:39:1F:9E:79:4B:04:80:25:30:B3:63:E1:21:DA:8A:30:43:BB:26:66:2F:EA:4D:CA:7F:C9:51:A4:B
Alias name: hellenicacademicandresearchinstitutionseccrootca2015
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3
Alias name: hellenicacademicandresearchinstitutionsrootca2011
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: hellenicacademicandresearchinstitutionsrootca2015
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
Alias name: hongkongpostrootca1
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: hongkongpostrootca3
  SHA1: 58:A2:D0:EC:20:52:81:5B:C1:F3:F8:64:02:24:4E:C2:8E:02:4B:02
  SHA256:
 5A:2F:C0:3F:0C:83:B0:90:BB:FA:40:60:4B:09:88:44:6C:76:36:18:3D:F9:84:6E:17:10:1A:44:7F:B8:EF:D
```

```
Alias name: identrustcommercialrootca1
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A
Alias name: identrustpublicsectorrootca1
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2
Alias name: isrgrootx1
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C
Alias name: izenpecom
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: keynectisrootca
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3
Alias name: microseceszignorootca2009
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert0.pem
  SHA1: 97:81:79:50:D8:1C:96:70:CC:34:D8:09:CF:79:44:31:36:7E:F4:74
  SHA256:
 A5:31:25:18:8D:21:10:AA:96:4B:02:C7:B7:C6:DA:32:03:17:08:94:E5:FB:71:FF:FB:66:67:D5:E6:81:0A:3
Alias name: mozillacert1.pem
  SHA1: 23:E5:94:94:51:95:F2:41:48:03:B4:D5:64:D2:A3:A3:F5:D8:8B:8C
  SHA256:
 B4:41:0B:73:E2:E6:EA:CA:47:FB:C4:2F:8F:A4:01:8A:F4:38:1D:C5:4C:FA:A8:44:50:46:1E:ED:09:45:4D:E
Alias name: mozillacert10.pem
  SHA1: 5F:3A:FC:0A:8B:64:F6:86:67:34:74:DF:7E:A9:A2:FE:F9:FA:7A:51
  SHA256:
 21:DB:20:12:36:60:BB:2E:D4:18:20:5D:A1:1E:E7:A8:5A:65:E2:BC:6E:55:B5:AF:7E:78:99:C8:A2:66:D9:2
Alias name: mozillacert100.pem
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: mozillacert101.pem
  SHA1: 99:A6:9B:E6:1A:FE:88:6B:4D:2B:82:00:7C:B8:54:FC:31:7E:15:39
  SHA256:
 62:F2:40:27:8C:56:4C:4D:D8:BF:7D:9D:4F:6F:36:6E:A8:94:D2:2F:5F:34:D9:89:A9:83:AC:EC:2F:FF:ED:5
```

```
Alias name: mozillacert102.pem
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: mozillacert103.pem
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B
Alias name: mozillacert104.pem
  SHA1: 4F:99:AA:93:FB:2B:D1:37:26:A1:99:4A:CE:7F:F0:05:F2:93:5D:1E
  SHA256:
 1C:01:C6:F4:DB:B2:FE:FC:22:55:8B:2B:CA:32:56:3F:49:84:4A:CF:C3:2B:7B:E4:B0:FF:59:9F:9E:8C:7A:F
Alias name: mozillacert105.pem
  SHA1: 77:47:4F:C6:30:E4:0F:4C:47:64:3F:84:BA:B8:C6:95:4A:8A:41:EC
  SHA256:
 F0:9B:12:2C:71:14:F4:A0:9B:D4:EA:4F:4A:99:D5:58:B4:6E:4C:25:CD:81:14:0D:29:C0:56:13:91:4C:38:4
Alias name: mozillacert106.pem
  SHA1: E7:A1:90:29:D3:D5:52:DC:0D:0F:C6:92:D3:EA:88:0D:15:2E:1A:6B
  SHA256:
 D9:5F:EA:3C:A4:EE:DC:E7:4C:D7:6E:75:FC:6D:1F:F6:2C:44:1F:0F:A8:BC:77:F0:34:B1:9E:5D:B2:58:01:5
Alias name: mozillacert107.pem
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C
Alias name: mozillacert108.pem
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: mozillacert109.pem
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: mozillacert11.pem
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: mozillacert110.pem
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: mozillacert111.pem
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
```

```
Alias name: mozillacert112.pem
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: mozillacert113.pem
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: mozillacert114.pem
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: mozillacert115.pem
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: mozillacert116.pem
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: mozillacert117.pem
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: mozillacert118.pem
  SHA1: 7E:78:4A:10:1C:82:65:CC:2D:E1:F1:6D:47:B4:40:CA:D9:0A:19:45
  SHA256:
 5F:0B:62:EA:B5:E3:53:EA:65:21:65:16:58:FB:B6:53:59:F4:43:28:0A:4A:FB:D1:04:D7:7D:10:F9:F0:4C:0
Alias name: mozillacert119.pem
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: mozillacert12.pem
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: mozillacert120.pem
  SHA1: DA:40:18:8B:91:89:A3:ED:EE:AE:DA:97:FE:2F:9D:F5:B7:D1:8A:41
  SHA256:
 CF:56:FF:46:A4:A1:86:10:9D:D9:65:84:B5:EE:B5:8A:51:0C:42:75:B0:E5:F9:4F:40:BB:AE:86:5E:19:F6:7
Alias name: mozillacert121.pem
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
```

```
Alias name: mozillacert122.pem
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: mozillacert123.pem
  SHA1: 2A:B6:28:48:5E:78:FB:F3:AD:9E:79:10:DD:6B:DF:99:72:2C:96:E5
  SHA256:
 07:91:CA:07:49:B2:07:82:AA:D3:C7:D7:BD:0C:DF:C9:48:58:35:84:3E:B2:D7:99:60:09:CE:43:AB:6C:69:2
Alias name: mozillacert124.pem
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: mozillacert125.pem
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: mozillacert126.pem
  SHA1: 25:01:90:19:CF:FB:D9:99:1C:B7:68:25:74:8D:94:5F:30:93:95:42
  SHA256:
 AF:8B:67:62:A1:E5:28:22:81:61:A9:5D:5C:55:9E:E2:66:27:8F:75:D7:9E:83:01:89:A5:03:50:6A:BD:6B:4
Alias name: mozillacert127.pem
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: mozillacert128.pem
  SHA1: A9:E9:78:08:14:37:58:88:F2:05:19:B0:6D:2B:0D:2B:60:16:90:7D
  SHA256:
 CA:2D:82:A0:86:77:07:2F:8A:B6:76:4F:F0:35:67:6C:FE:3E:5E:32:5E:01:21:72:DF:3F:92:09:6D:B7:9B:8
Alias name: mozillacert129.pem
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: mozillacert13.pem
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: mozillacert130.pem
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: mozillacert131.pem
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
```

```
Alias name: mozillacert132.pem
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: mozillacert133.pem
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: mozillacert134.pem
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2
Alias name: mozillacert135.pem
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: mozillacert136.pem
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: mozillacert137.pem
  SHA1: 4A:65:D5:F4:1D:EF:39:B8:B8:90:4A:4A:D3:64:81:33:CF:C7:A1:D1
  SHA256:
 BD:81:CE:3B:4F:65:91:D1:1A:67:B5:FC:7A:47:FD:EF:25:52:1B:F9:AA:4E:18:B9:E3:DF:2E:34:A7:80:3B:E
Alias name: mozillacert138.pem
  SHA1: E1:9F:E3:0E:8B:84:60:9E:80:9B:17:0D:72:A8:C5:BA:6E:14:09:BD
  SHA256:
 3F:06:E5:56:81:D4:96:F5:BE:16:9E:B5:38:9F:9F:2B:8F:F6:1E:17:08:DF:68:81:72:48:49:CD:5D:27:CB:6
Alias name: mozillacert139.pem
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: mozillacert14.pem
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: mozillacert140.pem
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: mozillacert141.pem
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
```

```
Alias name: mozillacert142.pem
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: mozillacert143.pem
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: mozillacert144.pem
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: mozillacert145.pem
  SHA1: 10:1D:FA:3F:D5:0B:CB:BB:9B:B5:60:0C:19:55:A4:1A:F4:73:3A:04
  SHA256:
 D4:1D:82:9E:8C:16:59:82:2A:F9:3F:CE:62:BF:FC:DE:26:4F:C8:4E:8B:95:0C:5F:F2:75:D0:52:35:46:95:A
Alias name: mozillacert146.pem
  SHA1: 21:FC:BD:8E:7F:6C:AF:05:1B:D1:B3:43:EC:A8:E7:61:47:F2:0F:8A
  SHA256:
 48:98:C6:88:8C:0C:FF:B0:D3:E3:1A:CA:8A:37:D4:E3:51:5F:F7:46:D0:26:35:D8:66:46:CF:A0:A3:18:5A:E
Alias name: mozillacert147.pem
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: mozillacert148.pem
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: mozillacert149.pem
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: mozillacert15.pem
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: mozillacert150.pem
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E
Alias name: mozillacert151.pem
  SHA1: AC:ED:5F:65:53:FD:25:CE:01:5F:1F:7A:48:3B:6A:74:9F:61:78:C6
  SHA256:
 7F:12:CD:5F:7E:5E:29:0E:C7:D8:51:79:D5:B7:2C:20:A5:BE:75:08:FF:DB:5B:F8:1A:B9:68:4A:7F:C9:F6:6
```

```
Alias name: mozillacert16.pem
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: mozillacert17.pem
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4
Alias name: mozillacert18.pem
  SHA1: 79:98:A3:08:E1:4D:65:85:E6:C2:1E:15:3A:71:9F:BA:5A:D3:4A:D9
  SHA256:
 44:04:E3:3B:5E:14:0D:CF:99:80:51:FD:FC:80:28:C7:C8:16:15:C5:EE:73:7B:11:1B:58:82:33:A9:B5:35:A
Alias name: mozillacert19.pem
  SHA1: B4:35:D4:E1:11:9D:1C:66:90:A7:49:EB:B3:94:BD:63:7B:A7:82:B7
  SHA256:
 C4:70:CF:54:7E:23:02:B9:77:FB:29:DD:71:A8:9A:7B:6C:1F:60:77:7B:03:29:F5:60:17:F3:28:BF:4F:6B:E
Alias name: mozillacert2.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: mozillacert20.pem
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: mozillacert21.pem
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: mozillacert22.pem
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: mozillacert23.pem
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: mozillacert24.pem
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6
Alias name: mozillacert25.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
```

```
Alias name: mozillacert26.pem
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: mozillacert27.pem
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: mozillacert28.pem
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: mozillacert29.pem
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: mozillacert3.pem
  SHA1: 87:9F:4B:EE:05:DF:98:58:3B:E3:60:D6:33:E7:0D:3F:FE:98:71:AF
  SHA256:
 39:DF:7B:68:2B:7B:93:8F:84:71:54:81:CC:DE:8D:60:D8:F2:2E:C5:98:87:7D:0A:AA:C1:2B:59:18:2B:03:1
Alias name: mozillacert30.pem
  SHA1: E7:B4:F6:9D:61:EC:90:69:DB:7E:90:A7:40:1A:3C:F4:7D:4F:E8:EE
  SHA256:
 A7:12:72:AE:AA:A3:CF:E8:72:7F:7F:B3:9F:0F:B3:D1:E5:42:6E:90:60:B0:6E:E6:F1:3E:9A:3C:58:33:CD:4
Alias name: mozillacert31.pem
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: mozillacert32.pem
  SHA1: 60:D6:89:74:B5:C2:65:9E:8A:0F:C1:88:7C:88:D2:46:69:1B:18:2C
  SHA256:
 B9:BE:A7:86:0A:96:2E:A3:61:1D:AB:97:AB:6D:A3:E2:1C:10:68:B9:7D:55:57:5E:D0:E1:12:79:C1:1C:89:3
Alias name: mozillacert33.pem
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: mozillacert34.pem
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: mozillacert35.pem
  SHA1: 2A:C8:D5:8B:57:CE:BF:2F:49:AF:F2:FC:76:8F:51:14:62:90:7A:41
  SHA256:
 92:BF:51:19:AB:EC:CA:D0:B1:33:2D:C4:E1:D0:5F:BA:75:B5:67:90:44:EE:0C:A2:6E:93:1F:74:4F:2F:33:C
```

```
Alias name: mozillacert36.pem
  SHA1: 23:88:C9:D3:71:CC:9E:96:3D:FF:7D:3C:A7:CE:FC:D6:25:EC:19:0D
  SHA256:
 32:7A:3D:76:1A:BA:DE:A0:34:EB:99:84:06:27:5C:B1:A4:77:6E:FD:AE:2F:DF:6D:01:68:EA:1C:4F:55:67:D
Alias name: mozillacert37.pem
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: mozillacert38.pem
  SHA1: CB:A1:C5:F8:B0:E3:5E:B8:B9:45:12:D3:F9:34:A2:E9:06:10:D3:36
  SHA256:
 A6:C5:1E:0D:A5:CA:0A:93:09:D2:E4:C0:E4:0C:2A:F9:10:7A:AE:82:03:85:7F:E1:98:E3:E7:69:E3:43:08:5
Alias name: mozillacert39.pem
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: mozillacert4.pem
  SHA1: E3:92:51:2F:0A:CF:F5:05:DF:F6:DE:06:7F:75:37:E1:65:EA:57:4B
  SHA256:
 0B:5E:ED:4E:84:64:03:CF:55:E0:65:84:84:40:ED:2A:82:75:8B:F5:B9:AA:1F:25:3D:46:13:CF:A0:80:FF:3
Alias name: mozillacert40.pem
  SHA1: 80:25:EF:F4:6E:70:C8:D4:72:24:65:84:FE:40:3B:8A:8D:6A:DB:F5
  SHA256:
 8D:A0:84:FC:F9:9C:E0:77:22:F8:9B:32:05:93:98:06:FA:5C:B8:11:E1:C8:13:F6:A1:08:C7:D3:36:B3:40:8
Alias name: mozillacert41.pem
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: mozillacert42.pem
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: mozillacert43.pem
  SHA1: F9:CD:0E:2C:DA:76:24:C1:8F:BD:F0:F0:AB:B6:45:B8:F7:FE:D5:7A
  SHA256:
 50:79:41:C7:44:60:A0:B4:70:86:22:0D:4E:99:32:57:2A:B5:D1:B5:BB:CB:89:80:AB:1C:B1:76:51:A8:44:D
Alias name: mozillacert44.pem
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: mozillacert45.pem
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
```

```
Alias name: mozillacert46.pem
  SHA1: 40:9D:4B:D9:17:B5:5C:27:B6:9B:64:CB:98:22:44:0D:CD:09:B8:89
  SHA256:
 EC:C3:E9:C3:40:75:03:BE:E0:91:AA:95:2F:41:34:8F:F8:8B:AA:86:3B:22:64:BE:FA:C8:07:90:15:74:E9:3
Alias name: mozillacert47.pem
  SHA1: 1B:4B:39:61:26:27:6B:64:91:A2:68:6D:D7:02:43:21:2D:1F:1D:96
  SHA256:
 E4:C7:34:30:D7:A5:B5:09:25:DF:43:37:0A:0D:21:6E:9A:79:B9:D6:DB:83:73:A0:C6:9E:B1:CC:31:C7:C5:2
Alias name: mozillacert48.pem
  SHA1: A0:A1:AB:90:C9:FC:84:7B:3B:12:61:E8:97:7D:5F:D3:22:61:D3:CC
  SHA256:
 0F:4E:9C:DD:26:4B:02:55:50:D1:70:80:63:40:21:4F:E9:44:34:C9:B0:2F:69:7E:C7:10:FC:5F:EA:FB:5E:3
Alias name: mozillacert49.pem
  SHA1: 61:57:3A:11:DF:0E:D8:7E:D5:92:65:22:EA:D0:56:D7:44:B3:23:71
  SHA256:
 B7:B1:2B:17:1F:82:1D:AA:99:0C:D0:FE:50:87:B1:28:44:8B:A8:E5:18:4F:84:C5:1E:02:B5:C8:FB:96:2B:2
Alias name: mozillacert5.pem
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: mozillacert50.pem
  SHA1: 8C:96:BA:EB:DD:2B:07:07:48:EE:30:32:66:A0:F3:98:6E:7C:AE:58
  SHA256:
 35:AE:5B:DD:D8:F7:AE:63:5C:FF:BA:56:82:A8:F0:0B:95:F4:84:62:C7:10:8E:E9:A0:E5:29:2B:07:4A:AF:B
Alias name: mozillacert51.pem
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: mozillacert52.pem
  SHA1: 8B:AF:4C:9B:1D:F0:2A:92:F7:DA:12:8E:B9:1B:AC:F4:98:60:4B:6F
  SHA256:
 E2:83:93:77:3D:A8:45:A6:79:F2:08:0C:C7:FB:44:A3:B7:A1:C3:79:2C:B7:EB:77:29:FD:CB:6A:8D:99:AE:A
Alias name: mozillacert53.pem
  SHA1: 7F:8A:B0:CF:D0:51:87:6A:66:F3:36:0F:47:C8:8D:8C:D3:35:FC:74
  SHA256:
 2D:47:43:7D:E1:79:51:21:5A:12:F3:C5:8E:51:C7:29:A5:80:26:EF:1F:CC:0A:5F:B3:D9:DC:01:2F:60:0D:1
Alias name: mozillacert54.pem
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: mozillacert55.pem
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
```

```
Alias name: mozillacert56.pem
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: mozillacert57.pem
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: mozillacert58.pem
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: mozillacert59.pem
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: mozillacert6.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: mozillacert60.pem
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: mozillacert61.pem
  SHA1: E0:B4:32:2E:B2:F6:A5:68:B6:54:53:84:48:18:4A:50:36:87:43:84
  SHA256:
 03:95:0F:B4:9A:53:1F:3E:19:91:94:23:98:DF:A9:E0:EA:32:D7:BA:1C:DD:9B:C8:5D:B5:7E:D9:40:0B:43:4
Alias name: mozillacert62.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: mozillacert63.pem
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert64.pem
  SHA1: 62:7F:8D:78:27:65:63:99:D2:7D:7F:90:44:C9:FE:B3:F3:3E:FA:9A
  SHA256:
 AB:70:36:36:5C:71:54:AA:29:C2:C2:9F:5D:41:91:16:3B:16:2A:22:25:01:13:57:D5:6D:07:FF:A7:BC:1F:7
Alias name: mozillacert65.pem
  SHA1: 69:BD:8C:F4:9C:D3:00:FB:59:2E:17:93:CA:55:6A:F3:EC:AA:35:FB
  SHA256:
 BC:23:F9:8A:31:3C:B9:2D:E3:BB:FC:3A:5A:9F:44:61:AC:39:49:4C:4A:E1:5A:9E:9D:F1:31:E9:9B:73:01:9
```

```
Alias name: mozillacert66.pem
  SHA1: DD:E1:D2:A9:01:80:2E:1D:87:5E:84:B3:80:7E:4B:B1:FD:99:41:34
  SHA256:
 E6:09:07:84:65:A4:19:78:0C:B6:AC:4C:1C:0B:FB:46:53:D9:D9:CC:6E:B3:94:6E:B7:F3:D6:99:97:BA:D5:9
Alias name: mozillacert67.pem
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: mozillacert68.pem
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: mozillacert69.pem
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: mozillacert7.pem
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: mozillacert70.pem
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: mozillacert71.pem
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: mozillacert72.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: mozillacert73.pem
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: mozillacert74.pem
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: mozillacert75.pem
  SHA1: D2:32:09:AD:23:D3:14:23:21:74:E4:0D:7F:9D:62:13:97:86:63:3A
  SHA256:
 08:29:7A:40:47:DB:A2:36:80:C7:31:DB:6E:31:76:53:CA:78:48:E1:BE:BD:3A:0B:01:79:A7:07:F9:2C:F1:7
```

```
Alias name: mozillacert76.pem
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: mozillacert77.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: mozillacert78.pem
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: mozillacert79.pem
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: mozillacert8.pem
  SHA1: 3E:2B:F7:F2:03:1B:96:F3:8C:E6:C4:D8:A8:5D:3E:2D:58:47:6A:0F
  SHA256:
 C7:66:A9:BE:F2:D4:07:1C:86:3A:31:AA:49:20:E8:13:B2:D1:98:60:8C:B7:B7:CF:E2:11:43:B8:36:DF:09:E
Alias name: mozillacert80.pem
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: mozillacert81.pem
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: mozillacert82.pem
  SHA1: 2E:14:DA:EC:28:F0:FA:1E:8E:38:9A:4E:AB:EB:26:C0:0A:D3:83:C3
  SHA256:
 FC:BF:E2:88:62:06:F7:2B:27:59:3C:8B:07:02:97:E1:2D:76:9E:D1:0E:D7:93:07:05:A8:09:8E:FF:C1:4D:1
Alias name: mozillacert83.pem
  SHA1: A0:73:E5:C5:BD:43:61:0D:86:4C:21:13:0A:85:58:57:CC:9C:EA:46
  SHA256:
 8C:4E:DF:D0:43:48:F3:22:96:9E:7E:29:A4:CD:4D:CA:00:46:55:06:1C:16:E1:B0:76:42:2E:F3:42:AD:63:0
Alias name: mozillacert84.pem
  SHA1: D3:C0:63:F2:19:ED:07:3E:34:AD:5D:75:0B:32:76:29:FF:D5:9A:F2
  SHA256:
 79:3C:BF:45:59:B9:FD:E3:8A:B2:2D:F1:68:69:F6:98:81:AE:14:C4:B0:13:9A:C7:88:A7:8A:1A:FC:CA:02:F
Alias name: mozillacert85.pem
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
```

```
Alias name: mozillacert86.pem
  SHA1: 74:2C:31:92:E6:07:E4:24:EB:45:49:54:2B:E1:BB:C5:3E:61:74:E2
  SHA256:
 E7:68:56:34:EF:AC:F6:9A:CE:93:9A:6B:25:5B:7B:4F:AB:EF:42:93:5B:50:A2:65:AC:B5:CB:60:27:E4:4E:7
Alias name: mozillacert87.pem
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: mozillacert88.pem
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: mozillacert89.pem
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: mozillacert9.pem
  SHA1: F4:8B:11:BF:DE:AB:BE:94:54:20:71:E6:41:DE:6B:BE:88:2B:40:B9
  SHA256:
 76:00:29:5E:EF:E8:5B:9E:1F:D6:24:DB:76:06:2A:AA:AE:59:81:8A:54:D2:77:4C:D4:C0:B2:C0:11:31:E1:B
Alias name: mozillacert90.pem
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: mozillacert91.pem
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: mozillacert92.pem
  SHA1: A3:F1:33:3F:E2:42:BF:CF:C5:D1:4E:8F:39:42:98:40:68:10:D1:A0
  SHA256:
 E1:78:90:EE:09:A3:FB:F4:F4:8B:9C:41:4A:17:D6:37:B7:A5:06:47:E9:BC:75:23:22:72:7F:CC:17:42:A9:1
Alias name: mozillacert93.pem
  SHA1: 31:F1:FD:68:22:63:20:EE:C6:3B:3F:9D:EA:4A:3E:53:7C:7C:39:17
  SHA256:
 C7:BA:65:67:DE:93:A7:98:AE:1F:AA:79:1E:71:2D:37:8F:AE:1F:93:C4:39:7F:EA:44:1B:B7:CB:E6:FD:59:9
Alias name: mozillacert94.pem
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: mozillacert95.pem
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
```

```
Alias name: mozillacert96.pem
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: mozillacert97.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: mozillacert98.pem
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7
Alias name: mozillacert99.pem
  SHA1: F1:7F:6F:B6:31:DC:99:E3:A3:C8:7F:FE:1C:F1:81:10:88:D9:60:33
  SHA256:
 97:8C:D9:66:F2:FA:A0:7B:A7:AA:95:00:D9:C0:2E:9D:77:F2:CD:AD:A6:AD:6B:A7:4A:F4:B9:1C:66:59:3C:5
Alias name: netlockaranyclassgoldfotanusitvany
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: networksolutionscertificateauthority
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: oistewisekeyglobalrootgaca
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: oistewisekeyglobalrootgbca
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D
Alias name: oistewisekeyglobalrootgcca
  SHA1: E0:11:84:5E:34:DE:BE:88:81:B9:9C:F6:16:26:D1:96:1F:C3:B9:31
  SHA256:
 85:60:F9:1C:36:24:DA:BA:95:70:B5:FE:A0:DB:E3:6F:F1:1A:83:23:BE:94:86:85:4F:B3:F3:4A:55:71:19:8
Alias name: quovadisrootca
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: quovadisrootca1g3
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7
```

```
Alias name: quovadisrootca2
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: quovadisrootca2g3
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4
Alias name: quovadisrootca3
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: quovadisrootca3g3
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4
Alias name: secomevrootca1
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: secomscrootca1
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: secomscrootca2
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: secomvalicertclass1ca
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: secureglobalca
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: securesignrootca11
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: securetrustca
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
```

```
Alias name: securitycommunicationrootca
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: securitycommunicationrootca2
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: soneraclass1ca
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3
Alias name: soneraclass2ca
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: soneraclass2rootca
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: sslcomevrootcertificationauthorityecc
  SHA1: 4C:DD:51:A3:D1:F5:20:32:14:B0:C6:C5:32:23:03:91:C7:46:42:6D
  SHA256:
 22:A2:C1:F7:BD:ED:70:4C:C1:E7:01:B5:F4:08:C3:10:88:0F:E9:56:B5:DE:2A:4A:44:F9:9C:87:3A:25:A7:C
Alias name: sslcomevrootcertificationauthorityrsar2
  SHA1: 74:3A:F0:52:9B:D0:32:A0:F4:4A:83:CD:D4:BA:A9:7B:7C:2E:C4:9A
  SHA256:
 2E:7B:F1:6C:C2:24:85:A7:BB:E2:AA:86:96:75:07:61:B0:AE:39:BE:3B:2F:E9:D0:CC:6D:4E:F7:34:91:42:5
Alias name: sslcomrootcertificationauthorityecc
  SHA1: C3:19:7C:39:24:E6:54:AF:1B:C4:AB:20:95:7A:E2:C3:0E:13:02:6A
  SHA256:
 34:17:BB:06:CC:60:07:DA:1B:96:1C:92:0B:8A:B4:CE:3F:AD:82:0E:4A:A3:0B:9A:CB:C4:A7:4E:BD:CE:BC:6
Alias name: sslcomrootcertificationauthorityrsa
  SHA1: B7:AB:33:08:D1:EA:44:77:BA:14:80:12:5A:6F:BD:A9:36:49:0C:BB
  SHA256:
 85:66:6A:56:2E:E0:BE:5C:E9:25:C1:D8:89:0A:6F:76:A8:7E:C1:6D:4D:7D:5F:29:EA:74:19:CF:20:12:3B:6
Alias name: staatdernederlandenevrootca
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5
Alias name: staatdernederlandenrootcag3
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2
```

```
Alias name: starfieldclass2ca
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: starfieldrootcertificateauthorityg2
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldrootg2ca
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldservicesrootcertificateauthorityg2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: starfieldservicesrootg2ca
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: swisssigngoldcag2
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssigngoldg2ca
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssignplatinumg2ca
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3
Alias name: swisssignsilvercag2
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: swisssignsilverg2ca
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: szafirrootca2
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F
```

```
Alias name: teliasonerarootcav1
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: thawtepersonalfreemailca
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8
Alias name: thawtepremiumserverca
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E
Alias name: thawteprimaryrootca
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: thawteprimaryrootcag2
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: thawteprimaryrootcag3
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: thawteserverca
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6
Alias name: trustcenterclass2caii
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: trustcenterclass4caii
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3
Alias name: trustcenteruniversalcai
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: trustcoreca1
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9
```

```
Alias name: trustcorrootcertca1
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5
Alias name: trustcorrootcertca2
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6
Alias name: trustisfpsrootca
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: ttelesecglobalrootclass2
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass2ca
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass3
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: ttelesecglobalrootclass3ca
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: tubitakkamusmsslkoksertifikasisurum1
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1
Alias name: twcaglobalrootca
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: twcarootcertificationauthority
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: ucaextendedvalidationroot
  SHA1: A3:A1:B0:6F:24:61:23:4A:E3:36:A5:C2:37:FC:A6:FF:DD:F0:D7:3A
  SHA256:
 D4:3A:F9:B3:54:73:75:5C:96:84:FC:06:D7:D8:CB:70:EE:5C:28:E7:73:FB:29:4E:B4:1E:E7:17:22:92:4D:2
```

```
Alias name: ucaglobalg2root
  SHA1: 28:F9:78:16:19:7A:FF:18:25:18:AA:44:FE:C1:A0:CE:5C:B6:4C:8A
  SHA256:
 9B:EA:11:C9:76:FE:01:47:64:C1:BE:56:A6:F9:14:B5:A5:60:31:7A:BD:99:88:39:33:82:E5:16:1A:A0:49:3
Alias name: usertrustecc
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustecccertificationauthority
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustrsa
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: usertrustrsacertificationauthority
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: utndatacorpsgcca
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: utnuserfirstclientauthemailca
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A
Alias name: utnuserfirsthardwareca
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: utnuserfirstobjectca
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8
Alias name: valicertclass2ca
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: verisignc1g1.pem
  SHA1: 90:AE:A2:69:85:FF:14:80:4C:43:49:52:EC:E9:60:84:77:AF:55:6F
  SHA256:
 D1:7C:D8:EC:D5:86:B7:12:23:8A:48:2C:E4:6F:A5:29:39:70:74:2F:27:6D:8A:B6:A9:E4:6E:E0:28:8F:33:5
```

```
Alias name: verisignc1g2.pem
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignc1g3.pem
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignc1g6.pem
  SHA1: 51:7F:61:1E:29:91:6B:53:82:FB:72:E7:44:D9:8D:C3:CC:53:6D:64
  SHA256:
 9D:19:0B:2E:31:45:66:68:5B:E8:A8:89:E2:7A:A8:C7:D7:AE:1D:8A:AD:DB:A3:C1:EC:F9:D2:48:63:CD:34:B
Alias name: verisignc2g1.pem
  SHA1: 67:82:AA:E0:ED:EE:E2:1A:58:39:D3:C0:CD:14:68:0A:4F:60:14:2A
  SHA256:
 BD:46:9F:F4:5F:AA:E7:C5:4C:CB:D6:9D:3F:3B:00:22:55:D9:B0:6B:10:B1:D0:FA:38:8B:F9:6B:91:8B:2C:E
Alias name: verisignc2g2.pem
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignc2g3.pem
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignc2g6.pem
  SHA1: 40:B3:31:A0:E9:BF:E8:55:BC:39:93:CA:70:4F:4E:C2:51:D4:1D:8F
  SHA256:
 CB:62:7D:18:B5:8A:D5:6D:DE:33:1A:30:45:6B:C6:5C:60:1A:4E:9B:18:DE:DC:EA:08:E7:DA:AA:07:81:5F:F
Alias name: verisignc3g1.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignc3g2.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignc3g3.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignc3g4.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
```

```
Alias name: verisignc3g5.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignc4g2.pem
  SHA1: 0B:77:BE:BB:CB:7A:A2:47:05:DE:CC:0F:BD:6A:02:FC:7A:BD:9B:52
  SHA256:
 44:64:0A:0A:0E:4D:00:0F:BD:57:4D:2B:8A:07:BD:B4:D1:DF:ED:3B:45:BA:AB:A7:6F:78:57:78:C7:01:19:6
Alias name: verisignc4g3.pem
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: verisignclass1ca
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2
Alias name: verisignclass1g2ca
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignclass1g3ca
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignclass2g2ca
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignclass2g3ca
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignclass3ca
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignclass3g2ca
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignclass3g3ca
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
```

```
Alias name: verisignclass3g4ca
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3g5ca
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignclass3publicprimarycertificationauthorityg4
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3publicprimarycertificationauthorityg5
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignroot.pem
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisigntsaca
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9
Alias name: verisignuniversalrootca
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisignuniversalrootcertificationauthority
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: xrampglobalca
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: xrampglobalcaroot
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
```

AWS WAF Zum Schutz Ihrer APIs verwenden

AWS WAF ist eine Firewall für Webanwendungen, die hilft, Webanwendungen und APIs vor Angriffen zu schützen. Sie ermöglicht es ihnen, eine Gruppe von Regeln (eine sogenannte Web-Zugriffskontrollliste oder Web-ACL) zum Zulassen, Blockieren oder Zählen von Webanforderungen basierend auf von Ihnen definierten anpassbaren Web-Sicherheitsregeln und Bedingungen zu konfigurieren. Weitere Informationen finden Sie unter So AWS WAF funktioniert es.

Sie können AWS WAF damit Ihre API Gateway REST API vor gängigen Web-Exploits wie SQL-Injection und Cross-Site Scripting (XSS) -Angriffen schützen. Diese können sich auf die Verfügbarkeit und Leistung der API auswirken, die Sicherheit gefährden oder übermäßige Ressourcen verbrauchen. Sie können beispielsweise Regeln zum Zulassen oder Blockieren von Anforderungen von angegebenen IP-Adressbereichen, Anforderungen von CIDR-Blocks, Anforderungen aus einem bestimmten Land oder einer bestimmten Region, Anforderungen mit bösartigem SQL-Code oder Anforderungen mit bösartigem Skript erstellen.

Sie können auch Regeln erstellen, die einer bestimmten Zeichenfolge oder einem regulären Ausdrucksmuster in HTTP-Headern, Methoden, Abfragezeichenfolgen, URI und dem Anforderungstext entsprechen (begrenzt auf die ersten 64 KB). Außerdem können Sie Regeln zum Blockieren von Angriffen von bestimmten Benutzeragenten, bösartigen Bots und Content Scrapers erstellen. Beispielsweise können Sie mit durchsatzbasierten Regeln die Anzahl der zulässigen Webanforderungen angeben, die von jeder Client-IP in einem sich anschließenden, fortlaufend aktualisierten 5-Minuten-Zeitraum zugelassen werden.

> **Important**
>
> AWS WAF ist Ihre erste Verteidigungslinie gegen Web-Exploits. Wenn auf einer API aktiviert AWS WAF ist, werden AWS WAF Regeln vor anderen Zugriffskontrollfunktionen wie Ressourcenrichtlinien, IAM-Richtlinien, Lambda-Autorisierern und Amazon CognitoCognito-Autorisierern ausgewertet. Wenn beispielsweise der Zugriff über einen CIDR-Block AWS WAF blockiert wird, den eine Ressourcenrichtlinie zulässt, hat dies AWS WAF Vorrang und die Ressourcenrichtlinie wird nicht bewertet.

Um die API AWS WAF zu aktivieren, müssen Sie wie folgt vorgehen:

1. Verwenden Sie die AWS WAF Konsole, das AWS SDK oder die CLI, um eine Web-ACL zu erstellen, die die gewünschte Kombination aus AWS WAF verwalteten Regeln und Ihren eigenen

benutzerdefinierten Regeln enthält. Weitere Informationen finden Sie unter Erste Schritte mit AWS WAF und Web Access Control Lists (Web-ACLs).

> **Important**
>
> API Gateway benötigt eine AWS WAFV2 Web-ACL für eine regionale Anwendung oder eine AWS WAF Classic Regional Web-ACL.

2. Ordnen Sie die AWS WAF Web-ACL einer API-Stufe zu. Sie können dies mithilfe der AWS WAF Konsole, des AWS SDK, der CLI oder der API Gateway Gateway-Konsole tun.

So verknüpfen Sie mithilfe der API Gateway Gateway-Konsole eine AWS WAF Web-ACL mit einer API-Gateway-API-Stufe

Gehen Sie wie folgt vor, um mit der API Gateway Gateway-Konsole eine AWS WAF Web-ACL einer vorhandenen API-Gateway-API-Stufe zuzuordnen:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine vorhandene API oder erstellen Sie eine neue.
3. Wählen Sie im Hauptnavigationsbereich Stages (Stufen) und anschließend eine Stufe aus.
4. Wählen Sie im Abschnitt Stage details (Stufendetails) die Option Edit (Bearbeiten) aus.
5. Wählen Sie unter Web Application Firewall (AWS WAF) Ihre Web-ACL aus.

   Wenn Sie verwenden AWS WAFV2, wählen Sie eine AWS WAFV2 Web-ACL für eine regionale Anwendung aus. Die Web-ACL und alle anderen AWS WAFV2 Ressourcen, die sie verwendet, müssen sich in derselben Region wie Ihre API befinden.

   Wenn Sie eine regionale Web-ACL verwenden AWS WAF Classic Regional, wählen Sie eine regionale Web-ACL aus.
6. Wählen Sie Änderungen speichern aus.

Ordnen Sie eine AWS WAF Web-ACL einer API-Gateway-API-Stufe zu, indem Sie den AWS CLI

Rufen Sie den associate-web-aclBefehl wie im folgenden Beispiel auf, AWS CLI um eine AWS WAFV2 Web-ACL für eine regionale Anwendung mit einer vorhandenen API-Gateway-API-Stufe zu verknüpfen:

```
aws wafv2 associate-web-acl \
--web-acl-arn  arn:aws:wafv2:{region}:111122223333:regional/webacl/test-cli/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
--resource-arn arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod
```

Rufen Sie den associate-web-aclBefehl AWS CLI auf, um eine AWS WAF Classic Regional Web-ACL mit einer vorhandenen API-Gateway-API-Stufe zu verknüpfen, wie im folgenden Beispiel gezeigt:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Ordnen Sie mithilfe der REST-API eine AWS WAF Web-ACL einer API-Stufe zu AWS WAF

Um mithilfe der AWS WAFV2 REST-API eine AWS WAFV2 Web-ACL für eine regionale Anwendung einer vorhandenen API-Gateway-API-Stufe zuzuordnen, verwenden Sie den AssociateWebACL-Befehl wie im folgenden Beispiel:

```
import boto3

wafv2 = boto3.client('wafv2')

wafv2.associate_web_acl(
    WebACLArn='arn:aws:wafv2:{region}:111122223333:regional/webacl/test/abc6aa3b-
fc33-4841-b3db-0ef3d3825b25',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Um mithilfe der AWS WAF REST-API eine AWS WAF Classic Regional Web-ACL einer vorhandenen API-Gateway-API-Stufe zuzuordnen, verwenden Sie den AssociateWebACL-Befehl wie im folgenden Beispiel:

```
import boto3

waf = boto3.client('waf-regional')

waf.associate_web_acl(
    WebACLId='aabc123a-fb4f-4fc6-becb-2b00831cadcf',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Drosseln der API-Anforderungen für einen besseren Durchsatz

Sie können Drosselungen und Kontingente für Ihre APIs konfigurieren, um sie davor zu schützen, von zu vielen Anfragen überfordert zu werden. Sowohl Drosselungen als auch Quoten werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere überprüft API Gateway die Rate und einen Burst von Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise `429 Too Many Requests` Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Target-Limits für individuelle API-Stufen oder -Methoden festlegen, um die Gesamtleistung in allen APIs in Ihrem Konto zu verbessern. Alternativ können Sie Nutzungspläne aktivieren, um Drosselungen für Client-Anforderungseinreichungen basierend auf angegebenen Anforderungsraten und -kontingente einzurichten.

Themen

- So werden Einstellungen für die Drosselungsgrenzen in API Gateway angewendet:
- Drosselung auf Kontoebene pro Region
- Konfiguration von Drosselungs-Targets auf API-Ebene und Stufenebene in einem Nutzungsplan
- Konfigurieren von Drosselungs-Targets auf Stufenebene
- Konfiguration von Drosselungs-Targets auf Methodenebene in einem Nutzungsplan

So werden Einstellungen für die Drosselungsgrenzen in API Gateway angewendet:

Bevor Sie Drosselungs- und Kontingenteinstellungen für Ihre API konfigurieren, sollten Sie wissen, wie sie von Amazon API Gateway angewendet werden.

Amazon API Gateway bietet vier grundlegende Arten von Drosselungseinstellungen:

- AWS Drosselungslimits gelten für alle Konten und Kunden in einer Region. Diese Limit-Einstellungen verhindern, dass Ihre API – und Ihr Konto – von zu vielen Anfragen überfordert wird. Diese Limits werden von einem Kunden festgelegt AWS und können von diesem nicht geändert werden.
- Grenzwerte pro Konto werden auf alle APIs in einem Konto in einer bestimmten Region angewendet. Das Ratenlimit auf Kontoebene kann auf Anforderung erhöht werden – höhere Limits sind mit APIs möglich, die kürzere Timeouts und kleinere Nutzlasten aufweisen. Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das AWS Supportcenter. Weitere Informationen finden Sie unter Kontingente und wichtige Hinweise. Beachten Sie, dass diese Grenzwerte nicht höher als die AWS Drosselungsgrenzen sein können.
- Die Drosselungsgrenzen pro API pro Stufe werden auf API-Methodenebene für eine Stufe angewendet. Sie können für alle Methoden dieselben Einstellungen konfigurieren oder für jede Methode unterschiedliche Drosseleinstellungen konfigurieren. Beachten Sie, dass diese Grenzwerte nicht höher als die AWS Drosselungsgrenzen sein dürfen.
- Client-basierte Drossel-Limits werden auf Clients angewendet, die mit Ihrem Nutzungsplan verknüpfte API-Schlüssel als Client-ID verwenden. Beachten Sie, dass diese Limits nicht höher sein dürfen als die Grenzwerte pro Konto.

Drosselungsbezogene API Gateway-Einstellungen werden in der folgenden Reihenfolge angewendet:

1. Client-basierte Drossel-Limits oder Drossel-Limits pro Methode, die Sie im Nutzungsplan für eine API-Stufe festlegen
2. Drossel-Limits pro Methode, die Sie für eine API-Stufe festlegen
3. Drosselung auf Kontoebene pro Region
4. AWS Regionale Drosselung

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) für alle APIs innerhalb eines AWS -Kontos pro Region. Die Steigerung (also die maximale Bucket-Größe) wird über alle APIs innerhalb eines AWS -Kontos pro Region wird ebenfalls beschränkt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von `429 Too Many Requests`-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter Kontingente und wichtige Hinweise.

Konfiguration von Drosselungs-Targets auf API-Ebene und Stufenebene in einem Nutzungsplan

In einem Nutzungsplan können Sie ein Drosselungs-Target pro Methode für alle Methoden auf API- oder Stufenebene festlegen. Sie können eine Drosselungsrate angeben, d. h. die Rate in Anfragen pro Sekunde, mit der Token zum Token-Bucket hinzugefügt werden. Sie können auch einen Drosselungs-Burst angeben, der der Kapazität des Token-Buckets entspricht.

Sie können die AWS CLI, die SDKs und die verwenden, AWS Management Console um einen Nutzungsplan zu erstellen. Weitere Informationen zum Erstellen eines Nutzungsplans finden Sie unter???.

Konfigurieren von Drosselungs-Targets auf Stufenebene

Sie können die AWS CLI, die SDKs und die verwenden, AWS Management Console um Drosselungsziele auf Stufenebene zu erstellen.

Weitere Informationen zur Verwendung von zum Erstellen von Drosselungszielen auf AWS Management Console Stufenebene finden Sie unter. ??? Weitere Informationen zur Verwendung der AWS CLI zum Erstellen von Drosselungszielen auf Stufenebene finden Sie unter create-stage.

Konfiguration von Drosselungs-Targets auf Methodenebene in einem Nutzungsplan

Sie können weitere Drosselungs-Targets auf Methodenebene in Usage Plans (Nutzungsplänen) festlegen, wie unter Erstellen eines Nutzungsplans dargestellt. In der API Gateway-Konsole werden diese durch die Angabe von `Resource=<resource>`, `Method=<method>` in der Einstellung Configure Method Throttling (Methodendrosselung konfigurieren) festgelegt. Für dieses Beispiel könnten Sie beispielsweise, angebenPetStore. `Resource=/pets` `Method=GET`

Private REST-APIs im Amazon API Gateway

Eine private API ist eine REST-API, die nur innerhalb einer Amazon VPC aufgerufen werden kann. Sie können über einen Schnittstellen-VPC-Endpunkt auf Ihre API zugreifen. Dabei handelt es sich um eine Endpunkt-Netzwerkschnittstelle, die Sie in Ihrer VPC erstellen. Schnittstellenendpunkte basieren auf einer TechnologieAWS PrivateLink, die es Ihnen ermöglicht, über private IP-Adressen privat auf AWS Dienste zuzugreifen.

Sie können es auch verwenden AWS Direct Connect , um eine Verbindung von einem lokalen Netzwerk zu Amazon VPC herzustellen und dann über diese Verbindung auf Ihre private API zuzugreifen. In allen Fällen verwendet der Datenverkehr zu Ihrer privaten API sichere Verbindungen und ist vom öffentlichen Internet isoliert. Der Verkehr verlässt das Amazon-Netzwerk nicht.

Bewährte Methoden für private APIs

Wir empfehlen Ihnen, bei der Erstellung Ihrer privaten API die folgenden bewährten Methoden zu verwenden:

- Verwenden Sie einen einzigen VPC-Endpunkt, um auf mehrere private APIs zuzugreifen. Dadurch wird die Anzahl der VPC-Endpoints reduziert, die Sie möglicherweise benötigen.
- Ordnen Sie Ihren VPC-Endpunkt Ihrer API zu. Dadurch wird ein Route 53-Alias-DNS-Eintrag erstellt und der Aufruf Ihrer privaten API vereinfacht.
- Aktivieren Sie privates DNS für Ihre VPC. Auf diese Weise können Sie Ihre API innerhalb einer VPC aufrufen, ohne den Host oder `x-apigw-api-id` Header übergeben zu müssen. Wenn Sie das private DNS nicht aktivieren, können Sie nur über das öffentliche DNS auf Ihre API zugreifen.
- Beschränken Sie den Zugriff auf Ihre private API auf bestimmte VPCs oder VPC-Endpunkte. Fügen Sie `aws:SourceVpc` der Ressourcenrichtlinie Ihrer API `aws:SourceVpce` Bedingungen hinzu, um den Zugriff einzuschränken.
- Für den sichersten Datenperimeter können Sie eine VPC-Endpunktrichtlinie erstellen. Dadurch wird der Zugriff auf die VPC-Endpunkte gesteuert, die Ihre private API aufrufen können.

Überlegungen zu privaten APIs

Die folgenden Überlegungen könnten sich auf Ihre Verwendung privater APIs auswirken:

- Es werden nur REST-APIs unterstützt.
- Benutzerdefinierte Domainnamen werden für private APIs nicht unterstützt.

- Sie können keine private API in eine Edge-optimierte API konvertieren.
- Private APIs unterstützen nur TLS 1.2. Frühere TLS-Versionen werden nicht unterstützt.
- Für VPC-Endpunkte für private APIs gelten die gleichen Einschränkungen wie für andere Schnittstellen-VPC-Endpunkte. Weitere Informationen finden Sie im AWS PrivateLink Handbuch unter Zugreifen auf einen AWS Dienst über einen Schnittstellen-VPC-Endpunkt. Weitere Hinweise zur Verwendung von API Gateway mit gemeinsam genutzten VPCs und Subnetzen finden Sie unter Gemeinsam genutzte Subnetze im AWS PrivateLink -Leitfaden.

Die nächsten Schritte für private APIs

Informationen zum Erstellen einer privaten API und zum Zuordnen eines VPC-Endpunkts finden Sie unter. the section called “Erstellen Sie eine private API” Eine Anleitung, in der Sie Abhängigkeiten AWS CloudFormation und eine private API in der erstellen AWS Management Console, finden Sie unterthe section called “Tutorial: Erstellen Sie eine private REST-API”.

Erstellen Sie eine private API

Bevor Sie eine private API erstellen, erstellen Sie zunächst einen VPC-Endpunkt für API Gateway. Als Nächstes erstellen Sie Ihre private API und fügen ihr eine Ressourcenrichtlinie hinzu. Optional können Sie Ihren VPC-Endpunkt mit Ihrer privaten API verknüpfen, um den Aufruf Ihrer API zu vereinfachen. Schließlich stellen Sie Ihre API bereit.

In den folgenden Verfahren wird beschrieben, wie Sie dies erreichen können. Sie können eine private REST-API mit dem AWS Management Console, AWS CLI oder einem AWS SDK erstellen.

Voraussetzungen

Um diese Schritte ausführen zu können, benötigen Sie eine vollständig konfigurierte VPC. Informationen zum Erstellen einer VPC finden Sie unter Nur VPC erstellen im Amazon VPC-Benutzerhandbuch. Um bei der Erstellung Ihrer VPC alle empfohlenen Schritte zu befolgen, aktivieren Sie privates DNS. Auf diese Weise können Sie Ihre API innerhalb einer VPC aufrufen, ohne den Host oder `x-apigw-api-id` Header übergeben zu müssen.

Um privates DNS zu aktivieren, müssen die `enableDnsHostnames` Attribute `enableDnsSupport` und Ihrer VPC auf eingestellt sein`true`. Weitere Informationen finden Sie unter DNS-Unterstützung in Ihrer VPC und Aktualisierung der DNS-Unterstützung für Ihre VPC.

Schritt 1: Erstellen Sie einen VPC-Endpunkt für API Gateway in Ihrer VPC

Das folgende Verfahren zeigt, wie Sie einen VPC-Endpunkt für API Gateway erstellen. Um einen VPC-Endpunkt für API Gateway zu erstellen, geben Sie die `execute-api` Domain für die Domain an, AWS-Region in der Sie Ihre private API erstellen. Die `execute-api` Domain ist der API Gateway Gateway-Komponentendienst für die API-Ausführung.

Wenn Sie Ihren VPC-Endpunkt für API Gateway erstellen, geben Sie die DNS-Einstellungen an. Wenn Sie privates DNS deaktivieren, können Sie nur über öffentliches DNS auf Ihre API zugreifen. Weitere Informationen finden Sie unter the section called “Problem: Ich kann von einem API-Gateway-VPC-Endpunkt aus keine Verbindung zu meiner öffentlichen API herstellen”.

AWS Management Console

So erstellen Sie einen VPC-Schnittstellen-Endpunkt für API Gateway

1. Melden Sie sich bei der Amazon VPC-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsmenü unter Virtual Private Cloud die Option Endpunkte.
3. Wählen Sie Endpunkt erstellen aus.
4. (Optional) Geben Sie für das Name-Tag einen Namen ein, um Ihren VPC-Endpunkt leichter identifizieren zu können.
5. Wählen Sie bei Service category (Servicekategorie) die Option AWS services (-Services) aus.
6. Geben **`execute-api`** Sie unter Dienste in der Suchleiste Folgendes ein. Wählen Sie dann den API Gateway Gateway-Dienstendpunkt aus, auf AWS-Region dem Sie Ihre API erstellen möchten. Der Dienstname sollte wie folgt aussehen `com.amazonaws.us-east-1.execute-api` und der Typ sollte Interface sein.
7. Wählen Sie für VPC die VPC aus, in der Sie den Endpunkt erstellen möchten.
8. (Optional) Um „Privaten DNS-Namen aktivieren“ zu deaktivieren, wählen Sie „Zusätzliche Einstellungen“ und deaktivieren Sie dann die Option „Privaten DNS-Namen aktivieren“.
9. Wählen Sie für Subnetze die Availability Zones aus, in denen Sie die Endpunkt-Netzwerkschnittstellen erstellt haben. Zur Verbesserung der Verfügbarkeit Ihrer API wählen Sie mehrere Subnetze aus.
10. Wählen Sie für Security group (Sicherheitsgruppe) die Sicherheitsgruppe aus, die den VPC-Endpunktnetzwerkschnittstellen zugeordnet werden soll.

Die von Ihnen ausgewählte Sicherheitsgruppe muss so konfiguriert sein, dass eingehender HTTPS-Datenverkehr über TCP-Port 443 von einem IP-Adressbereich in Ihrer VPC oder einer anderen Sicherheitsgruppe in Ihrer VPC zugelassen wird.

11. Führen Sie für Policy einen der folgenden Schritte aus:
    - Wenn Sie Ihre private API nicht erstellt haben oder keine benutzerdefinierte VPC-Endpunktrichtlinie konfigurieren möchten, wählen Sie Vollzugriff.
    - Wenn Sie bereits eine private API erstellt haben und eine benutzerdefinierte VPC-Endpunktrichtlinie konfigurieren möchten, können Sie eine benutzerdefinierte VPC-Endpunktrichtlinie eingeben. Weitere Informationen finden Sie unter the section called “Verwenden von VPC-Endpunktrichtlinien für private-APIs”.

    Sie können die VPC-Endpunktrichtlinie aktualisieren, nachdem Sie Ihren VPC-Endpunkt erstellt haben. Weitere Informationen finden Sie unter Aktualisieren einer VPC-Endpunktrichtlinie.
12. Wählen Sie Endpunkt erstellen aus.
13. Kopieren Sie die resultierende VPC-Endpunkt-ID, da Sie sie in future Schritten verwenden könnten.

AWS CLI

Der folgende create-vpc-endpointBefehl kann verwendet werden, um einen VPC-Endpunkt zu erstellen:

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.execute-api \
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d
```

Kopieren Sie die resultierende VPC-Endpunkt-ID, da Sie sie in future Schritten verwenden könnten.

Schritt 2: Erstellen einer privaten API

Nachdem Sie Ihren VPC-Endpunkt erstellt haben, erstellen Sie eine private REST-API. Das folgende Verfahren zeigt, wie Sie eine private API erstellen.

AWS Management Console

Erstellen einer privaten API

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Create API (API erstellen) aus.
3. Wählen Sie unter REST-API die Option Build (Erstellen) aus.
4. Geben Sie unter Name einen Namen ein.
5. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
6. Wählen Sie für API endpoint type (API-Endpunkttyp) die Option Private (Privat) aus.
7. (Optional) Geben Sie für VPC-Endpunkt-IDs eine VPC-Endpunkt-ID ein.

   Wenn Sie Ihrer privaten API eine VPC-Endpunkt-ID zuordnen, können Sie Ihre API von Ihrer VPC aus aufrufen, ohne einen `Host` Header zu überschreiben oder eine zu übergeben. `x-apigw-api-id header` Weitere Informationen finden Sie unter. the section called “(Optional) Ordnen Sie einen VPC-Endpunkt einer privaten API zu oder trennen Sie die Zuordnung”
8. Wählen Sie Create API (API erstellen) aus.

Nachdem Sie die vorherigen Schritte abgeschlossen haben, können Sie den Anweisungen unter folgen, the section called “Erste Schritte mit der REST-API-Konsole” um Methoden und Integrationen für diese API einzurichten, aber Sie können Ihre API nicht bereitstellen. Um Ihre API bereitzustellen, folgen Sie Schritt 3 und fügen Sie Ihrer API eine Ressourcenrichtlinie hinzu.

AWS CLI

Der folgende update-rest-apiBefehl zeigt, wie Sie eine private API erstellen:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Private)' \
        --description 'Simple private PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["PRIVATE"] }'
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
{
```

```
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple private PetStore API",
    "endpointConfiguration": {
        "types": "PRIVATE"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Private)"
}
```

Nachdem Sie die vorherigen Schritte abgeschlossen haben, können Sie den Anweisungen unter folgen, the section called “Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI” um Methoden und Integrationen für diese API einzurichten, aber Sie können Ihre API nicht bereitstellen. Um Ihre API bereitzustellen, folgen Sie Schritt 3 und fügen Sie Ihrer API eine Ressourcenrichtlinie hinzu.

SDK JavaScript v3

Das folgende Beispiel zeigt, wie Sie mithilfe des AWS SDK für JavaScript Version 3 eine private API erstellen:

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
const apig = new APIGatewayClient({region:"us-east-1"});

const input = { // CreateRestApiRequest
  name: "Simple PetStore (JavaScript v3 SDK, private)", // required
  description: "Demo private API created using the AWS SDK for JavaScript v3",
  version: "0.00.001",
  endpointConfiguration: { // EndpointConfiguration
    types: [ "PRIVATE"],
  },
};

export const handler = async (event) => {
const command = new CreateRestApiCommand(input);
try {
  const result = await apig.send(command);
  console.log(result);
} catch (err){
  console.error(err)
 }
};
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
{
  apiKeySource: 'HEADER',
  createdDate: 2024-04-03T17:56:36.000Z,
  description: 'Demo private API created using the AWS SDK for JavaScript v3',
  disableExecuteApiEndpoint: false,
  endpointConfiguration: { types: [ 'PRIVATE' ] },
  id: 'abcd1234',
  name: 'Simple PetStore (JavaScript v3 SDK, private)',
  rootResourceId: 'efg567',
  version: '0.00.001'
}
```

Nachdem Sie die vorherigen Schritte abgeschlossen haben, können Sie den Anweisungen unter folgen, the section called “Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI” um Methoden und Integrationen für diese API einzurichten, aber Sie können Ihre API nicht bereitstellen. Um Ihre API bereitzustellen, folgen Sie Schritt 3 und fügen Sie Ihrer API eine Ressourcenrichtlinie hinzu.

Python SDK

Das folgende Beispiel zeigt, wie Sie mithilfe des AWS SDK für Python eine private API erstellen:

```
import json
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

def lambda_handler(event, context):
    try:
      result = apig.create_rest_api(
      name='Simple PetStore (Python SDK, private)',
      description='Demo private API created using the AWS SDK for Python',
      version='0.00.001',
      endpointConfiguration={
          'types': [
             'PRIVATE',
          ],
      },
      )
    except botocore.exceptions.ClientError as error:
            logger.exception("Couldn't create private API %s.", error)
```

```
            raise
    attribute=["id", "name", "description", "createdDate", "version",
 "apiKeySource", "endpointConfiguration"]
    filtered_data ={key:result[key] for key in attribute}
    result = json.dumps(filtered_data, default=str, sort_keys='true')
    return result
```

Bei einem erfolgreichen Aufruf sieht die zurückgegebene Ausgabe in etwa wie folgt aus:

```
"{\"apiKeySource\": \"HEADER\", \"createdDate\": \"2024-04-03 17:27:05+00:00\",
 \"description\": \"Demo private API created using the AWS SDK for \",
 \"endpointConfiguration\": {\"types\": [\"PRIVATE\"]}, \"id\": \"abcd1234\", \"name
\": \"Simple PetStore (Python SDK, private)\", \"version\": \"0.00.001\"}"
```

Nachdem Sie die vorherigen Schritte abgeschlossen haben, können Sie den Anweisungen unter folgen, the section called “Tutorial: Erstellen Sie eine Edge-optimierte API mithilfe von AWS SDKs oder AWS CLI” um Methoden und Integrationen für diese API einzurichten, aber Sie können Ihre API nicht bereitstellen. Um Ihre API bereitzustellen, folgen Sie Schritt 3 und fügen Sie Ihrer API eine Ressourcenrichtlinie hinzu.

Schritt 3: Richten Sie eine Ressourcenrichtlinie für eine private API ein

Ihre aktuelle private API ist für alle VPCs nicht zugänglich. Verwenden Sie eine Ressourcenrichtlinie, um Ihren VPCs und VPC-Endpunkten Zugriff auf Ihre privaten APIs zu gewähren. Sie können in jedem AWS Konto Zugriff auf einen VPC-Endpunkt gewähren.

Ihre Ressourcenrichtlinie sollte `aws:SourceVpce` Bedingungen zur Beschränkung des Zugriffs enthaltenaws:SourceVpc. Wir empfehlen, dass Sie bestimmte VPCs und VPC-Endpoints identifizieren und keine Ressourcenrichtlinie erstellen, die den Zugriff für alle VPCs und VPC-Endpoints ermöglicht.

Das folgende Verfahren zeigt, wie Sie eine Ressourcenrichtlinie an Ihre API anhängen.

AWS Management Console

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine REST-API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.

4. Wählen Sie Richtlinie erstellen aus.
5. Wählen Sie Vorlage auswählen und dann Quell-VPC aus.
6. Ersetzen Sie `{{vpceID}}` (einschließlich der geschweiften Klammern) durch Ihre VPC-Endpunkt-ID.
7. Wählen Sie Änderungen speichern aus.

AWS CLI

Der folgende update-rest-apiBefehl zeigt, wie eine Ressourcenrichtlinie an eine bestehende API angehängt wird:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

Möglicherweise möchten Sie auch kontrollieren, welche Ressourcen Zugriff auf Ihren VPC-Endpunkt haben. Um zu kontrollieren, welche Ressourcen Zugriff auf Ihren VPC-Endpunkt haben, fügen Sie Ihrem VPC-Endpunkt eine Endpunktrichtlinie hinzu. Weitere Informationen finden Sie unter the section called “Verwenden von VPC-Endpunktrichtlinien für private-APIs”.

(Optional) Ordnen Sie einen VPC-Endpunkt einer privaten API zu oder trennen Sie die Zuordnung

Wenn Sie Ihrer privaten API einen VPC-Endpunkt zuordnen, generiert API Gateway einen neuen Route 53-Alias-DNS-Eintrag. Sie können diesen Datensatz verwenden, um Ihre privaten APIs genauso aufzurufen, wie Sie es mit Ihren öffentlichen APIs tun, ohne einen Header zu überschreiben oder einen `Host` Header zu übergeben. `x-apigw-api-id`

Die generierte Basis-URL liegt im folgenden Format vor:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (AWS Management Console)

Sie können Ihrer privaten API einen VPC-Endpunkt zuordnen, wenn Sie ihn erstellen oder nachdem er erstellt wurde. Das folgende Verfahren zeigt, wie Sie einen VPC-Endpunkt mit einer zuvor erstellten API verknüpfen.

So verknüpfen Sie einen VPC-Endpunkt mit einer privaten API

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre private API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Bearbeiten Sie Ihre Ressourcenrichtlinie, um Anrufe von Ihrem zusätzlichen VPC-Endpunkt zuzulassen.
5. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
6. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
7. Wählen Sie für VPC-Endpunkt-IDs zusätzliche VPC-Endpunkt-IDs aus.
8. Wählen Sie Speichern.
9. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Dissociate a VPC endpoint (AWS Management Console)

Die Zuordnung eines VPC-Endpunkts zu einer privaten REST-API entfernen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre private API aus.
3. Wählen Sie im Hauptnavigationsbereich Ressourcenrichtlinie.
4. Bearbeiten Sie Ihre Ressourcenrichtlinie, um Erwähnungen des VPC-Endpunkts zu entfernen, den Sie von Ihrer privaten API trennen möchten.
5. Wählen Sie im Hauptnavigationsbereich API-Einstellungen.
6. Wählen Sie im Abschnitt API-Details die Option Bearbeiten aus.
7. Für VPC-Endpunkt-IDs klicken Sie auf das X, um den VPC-Endpunkt zu trennen.
8. Wählen Sie Speichern.
9. Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Associate a VPC endpoint (AWS CLI)

Der folgende create-rest-apiBefehl zeigt, wie VPC-Endpunkte zum Zeitpunkt der API-Erstellung verknüpft werden:

```
aws apigateway create-rest-api \
    --name Petstore \
    --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
    --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee"
        ]
    },
    "id": "u67n3ov968",
    "createdDate": 1565718256,
    "name": "Petstore"
}
```

Der folgende update-rest-apiBefehl zeigt, wie Sie VPC-Endpunkte mit einer API verknüpfen, die Sie bereits erstellt haben:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
    --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
```

```
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Disassociate a VPC endpoint (AWS CLI)

Der folgende update-rest-apiBefehl zeigt, wie Sie einen VPC-Endpunkt von einer privaten API trennen:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='remove',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-0393a628149c867ee'" \
    --region us-west-2
```

Die Ausgabe sieht wie folgt aus:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Sie müssen Ihre API erneut bereitstellen, damit die Änderungen wirksam werden.

Schritt 4: Stellen Sie eine private API bereit

Um Ihre API bereitzustellen, erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Phase. Das folgende Verfahren zeigt, wie Sie Ihre private API bereitstellen.

AWS Management Console

Um eine private API bereitzustellen

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

AWS CLI

Der folgende Befehl create-deployment zeigt, wie eine private API bereitgestellt wird:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \
  --stage-name test \
  --stage-description 'Private API test stage' \
  --description 'First deployment'
```

Beheben Sie Fehler bei Ihrer privaten API

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Erstellung einer privaten API auftreten können.

Problem: Ich kann von einem API-Gateway-VPC-Endpunkt aus keine Verbindung zu meiner öffentlichen API herstellen

Wenn Sie Ihre VPC erstellen, können Sie die DNS-Einstellungen konfigurieren. Wir empfehlen, dass Sie privates DNS für Ihre VPC aktivieren. Wenn Sie sich dafür entscheiden, privates DNS zu deaktivieren, können Sie nur über öffentliches DNS auf Ihre API zugreifen.

Wenn Sie privates DNS aktivieren, können Sie von Ihrem VPC-Endpunkt aus nicht auf den Standardendpunkt einer öffentlichen API-Gateway-API zugreifen. Sie können mit einem benutzerdefinierten Domainnamen auf eine API zugreifen.

Wenn Sie einen benutzerdefinierten regionalen Domainnamen erstellen, verwenden Sie einen Aliaseintrag vom Typ A. Wenn Sie einen Edge-optimierten benutzerdefinierten Domainnamen erstellen, gibt es keine Einschränkungen für Ihren Datensatztyp. Sie können auf diese öffentlichen APIs zugreifen, wenn privates DNS aktiviert ist. Weitere Informationen finden Sie unter Problem: Ich stelle von einem API-Gateway-VPC-Endpunkt aus eine Verbindung zu meiner öffentlichen API her.

Problem: Meine API gibt zurück **`{"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:********/****/****/"}`**

Wenn Sie in Ihrer Ressourcenrichtlinie den Principal auf einen AWS Prinzipal festlegen, wie zum Beispiel:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            ""Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:role/developer",
                    "arn:aws:iam::account-id:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        ...
        }
```

Sie müssen die `AWS_IAM` Autorisierung für jede Methode in Ihrer API verwenden, andernfalls gibt Ihre API die vorherige Fehlermeldung zurück. Weitere Anweisungen zum Aktivieren der `AWS_IAM` Autorisierung für eine Methode finden Sie unterthe section called “Methoden”.

Problem: Ich kann nicht feststellen, ob mein VPC-Endpunkt mit meiner API verknüpft ist

Wenn Sie einen VPC-Endpunkt mit Ihrer privaten API verknüpfen oder trennen, müssen Sie Ihre API erneut bereitstellen. Der Aktualisierungsvorgang kann aufgrund der DNS-Verbreitung einige Minuten dauern. Während dieser Zeit ist Ihre API verfügbar, aber die DNS-Verbreitung für die neu generierten DNS-URLs wird möglicherweise noch ausgeführt. Wenn Ihre neuen URLs nach einigen Minuten nicht im DNS aufgelöst werden, empfehlen wir Ihnen, Ihre API erneut bereitzustellen.

Rufen Sie eine private API auf

Sie können eine private API nur innerhalb einer VPC aufrufen. Ihre private API muss über eine Ressourcenrichtlinie verfügen, die es bestimmten VPCs und VPC-Endpunkten ermöglicht, Ihre API aufzurufen.

Sie können Ihre private API auf folgende Weise aufrufen:

- Rufen Sie Ihre API mit einem Route53-Alias auf. Dies ist nur verfügbar, wenn Sie Ihren VPC-Endpunkt mit Ihrer API verknüpft haben. Weitere Informationen finden Sie unter the section called “(Optional) Ordnen Sie einen VPC-Endpunkt einer privaten API zu oder trennen Sie die Zuordnung”.
- Rufen Sie Ihre API mit privatem DNS auf. Dies ist nur verfügbar, wenn Sie privates DNS für Ihre VPC aktiviert haben.
- Rufen Sie Ihre API auf mit. AWS Direct Connect
- Rufen Sie Ihre API mit endpunktspezifischen öffentlichen DNS-Hostnamen auf.

Um Ihre private API mit einem DNS-Namen aufzurufen, müssen Sie die DNS-Namen für Ihre API identifizieren. Das folgende Verfahren zeigt, wie Sie Ihre DNS-Namen finden.

AWS Management Console

Um die DNS-Namen zu finden

1. Melden Sie sich bei der Amazon VPC-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Hauptnavigationsbereich Endpoints und dann Ihren Schnittstellen-VPC-Endpunkt für API Gateway aus.

3. Im Detailbereich sehen Sie fünf Werte im Feld DNS-Namen. Die ersten drei sind die öffentlichen DNS-Namen für Ihre API. Die anderen beiden sind die privaten DNS-Namen dafür.

AWS CLI

Verwenden Sie den folgenden describe-vpc-endpointsBefehl, um Ihre DNS-Werte aufzulisten.

```
aws ec2 describe-vpc-endpoints --filters vpc-endpoint-id=vpce-01234567abcdef012
```

Die ersten drei sind die öffentlichen DNS-Namen für Ihre API. Die anderen beiden sind die privaten DNS-Namen dafür.

Rufen Sie eine private API mit einem Route53-Alias auf

Sie können einen VPC-Endpunkt mit Ihrer privaten API verknüpfen oder die Zuordnung aufheben. Weitere Informationen finden Sie unter the section called “(Optional) Ordnen Sie einen VPC-Endpunkt einer privaten API zu oder trennen Sie die Zuordnung”.

Nachdem Sie Ihre VPC-Endpoints mit Ihrer privaten API verknüpft haben, können Sie die folgende Basis-URL verwenden, um die API aufzurufen:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die `GET /pets` Methode für die `test` Phase eingerichtet haben und Ihre REST-API-ID und Ihre VPC-Endpunkt-ID und Ihre Region waren`us-west-2`, können Sie Ihre API wie folgt aufrufen: `01234567ab` `vpce-01234567abcdef012`

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/
test/pets
```

Rufen Sie eine private API mit privaten DNS-Namen auf

Wenn Sie privates DNS aktiviert haben, können Sie mit dem folgenden privaten DNS-Namen auf Ihre private API zugreifen:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

Die Basis-URL zum Aufrufen der API besitzt das folgende Format:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die `GET /pets` Methode für die `test` Phase eingerichtet haben und Ihre REST-API-ID `01234567ab` und Ihre Region lauten`us-west-2`, können Sie Ihre private API aufrufen, indem Sie die folgende URL in einen Browser eingeben:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Alternativ können Sie den folgenden cURL-Befehl verwenden, um Ihre private API aufzurufen:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

> **Warning**
>
> Wenn Sie privates DNS für Ihren VPC-Endpunkt aktivieren, können Sie auf den Standardendpunkt für öffentliche APIs zugreifen. Weitere Informationen finden Sie unter Warum kann ich von einem API Gateway-VPC-Endpunkt keine Verbindung zu meiner öffentlichen API herstellen?.

Rufen Sie eine private API auf mit AWS Direct Connect

Sie können AWS Direct Connect damit eine dedizierte private Verbindung von einem lokalen Netzwerk zu Amazon VPC herstellen und über diese Verbindung mithilfe öffentlicher DNS-Namen auf Ihren privaten API-Endpunkt zugreifen.

Sie können auch private DNS-Namen verwenden, um von einem lokalen Netzwerk aus auf Ihre private API zuzugreifen, indem Sie einen Amazon Route 53 Resolver eingehenden Endpunkt einrichten und an diesen alle DNS-Abfragen des privaten DNS aus Ihrem Remote-Netzwerk weiterleiten. Weitere Informationen finden Sie unter Weiterleiten von eingehenden DNS-Abfragen an Ihre VPCs im Entwicklerhandbuch für Amazon Route 53.

Rufen Sie eine private API mit endpunktspezifischen öffentlichen DNS-Hostnamen auf

Sie können auf Ihre private API mit endpunktspezifischen DNS-Hostnamen zugreifen. Dabei handelt es sich um öffentliche DNS-Hostnamen mit der VPC-Endpunkt-ID oder der API-ID für Ihre private API.

Die generierte Basis-URL liegt im folgenden Format vor:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Wenn Sie beispielsweise die `GET /pets` Methode für die `test` Phase eingerichtet haben und Ihre REST-API-ID`abc1234`, ihr öffentlicher DNS-Hostname und Ihre Region lautete`vpce-def-01234567`, könnten Sie Ihre private API mit ihrer vPCe-ID aufrufen`us-west-2`, indem Sie den `Host` Header in einem cURL-Befehl verwenden:

```
curl -v https://vpce-def-01234567.execute-api.us-west-2.vpce.amazonaws.com/test/pets -H
 'Host: abc1234.execute-api.us-west-2.amazonaws.com'
```

Alternativ können Sie Ihre private API über ihre API-ID aufrufen, indem Sie den `x-apigw-api-id` Header in einem cURL-Befehl im folgenden Format verwenden:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -
H 'x-apigw-api-id:{api-id}'
```

Überwachung von REST-APIs

In diesem Abschnitt erfahren Sie, wie Sie Ihre API mithilfe von CloudWatch Metriken, CloudWatch Logs, Firehose und AWS X-Rayüberwachen können. Durch die Kombination von CloudWatch Ausführungsprotokollen und CloudWatch Metriken können Sie Fehler und Ausführungsspuren protokollieren und die Leistung Ihrer API überwachen. Möglicherweise möchten Sie auch API-Aufrufe an Firehose protokollieren. Sie können es auch verwenden AWS X-Ray , um Aufrufe über die Downstream-Dienste zu verfolgen, aus denen Ihre API besteht.

> **Note**
>
> API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:
>
> - 413 Request Entity Too Large-Fehler
> - Übermäßige 429 Too Many Requests-Fehler
> - Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
> - Fehler der Serie 500, die durch interne Ausfälle verursacht werden

API Gateway generiert beim Testen einer REST-API-Methode keine Protokolle und Metriken. Die CloudWatch Einträge werden simuliert. Weitere Informationen finden Sie unter the section called “Testen einer REST-API-Methode mithilfe der Konsole”.

Themen

- Überwachung der REST-API-Ausführung mit CloudWatch Amazon-Metriken
- CloudWatch Protokollierung für eine REST-API in API Gateway einrichten
- Protokollieren von API-Aufrufen bei Amazon Data Firehose
- Ablaufverfolgung von Benutzeranforderungen an REST-APIs mithilfe von X-Ray

Überwachung der REST-API-Ausführung mit CloudWatch Amazon-Metriken

Sie können die API-Ausführung überwachen CloudWatch, indem Sie die Rohdaten von API Gateway sammeln und in lesbare near-real-time Metriken umwandeln. Diese Statistiken werden für einen Zeitraum von 15 Monaten aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt wird. Standardmäßig werden API-Gateway-Metrikdaten automatisch innerhalb von CloudWatch einer Minute gesendet. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch? im CloudWatch Amazon-Benutzerhandbuch.

Die von API Gateway gemeldeten Metriken bieten Informationen, die Sie auf unterschiedliche Weise analysieren können. Die folgende Liste zeigt einige häufige Verwendungszwecke für die Metriken, die Vorschläge für den Einstieg sind:

- Überwachen Sie die IntegrationLatencyMetriken, um die Reaktionsfähigkeit des Backends zu messen.
- Überwachen Sie die Latency-Metriken, um die allgemeine Reaktionsfähigkeit Ihrer API-Aufrufe zu messen.
- Überwachen Sie die CacheHitCountund CacheMissCount-Metriken, um die Cache-Kapazitäten zu optimieren und die gewünschte Leistung zu erzielen.

Themen

- Amazon API Gateway-Dimensionen und -Metriken

- CloudWatch Metriken mit dem API-Dashboard in API Gateway anzeigen
- API Gateway Gateway-Metriken in der CloudWatch Konsole anzeigen
- API Gateway Gateway-Protokollereignisse in der CloudWatch Konsole anzeigen
- Überwachungstools in AWS

Amazon API Gateway-Dimensionen und -Metriken

Die Metriken und Dimensionen, die API Gateway an Amazon sendet, CloudWatch sind unten aufgeführt. Weitere Informationen finden Sie unter Überwachung der REST-API-Ausführung mit CloudWatch Amazon-Metriken.

API Gateway-Metriken

Amazon API Gateway sendet CloudWatch jede Minute Metrikdaten an .

Der `AWS/ApiGateway`-Namespace enthält die folgenden Metriken.

Metrik	Beschreibung
4XXError	Die Anzahl der erfassten clientseitigen Fehler innerhalb eines bestimmten Zeitraums. API Gateway zählt geänderte Gateway-Antwortsta tuscodes als 4XXError-Fehler. Die Statistik `Sum` repräsentiert diese Metrik, also die Gesamtzahl der 4XXError-Fehler in dem angegeben en Zeitraum. Die Statistik `Average` repräsentiert die 4XXError-Fehlerrate, also die Gesamtzahl der 4XXError-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
5XXError	Die Anzahl der erfassten serverseitigen Fehler innerhalb eines bestimmten Zeitraums

Metrik	Beschreibung
	Die Statistik `Sum` repräsentiert diese Metrik, also die Gesamtzahl der 5XXError-Fehler in dem angegeben en Zeitraum. Die Statistik `Average` repräsentiert die 5XXError-Fehlerrate, also die Gesamtzahl der 5XXError-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
CacheHitCount	Die Anzahl aller vom API-Cache innerhalb eines bestimmten Zeitraums bedienten Anforderungen Die Statistik `Sum` repräsentiert diese Metrik, also die Gesamtzahl der Cache-Zugriffe innerhalb des angegebenen Zeitraums. Die Statistk `Average` stellt die Cache-Zugriffsrate, also die Gesamtzahl der Cache-Zugriffe geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums, dar. Der Nenner entspricht der Count-Metrik (unten). Unit: Count
CacheMissCount	Die Anzahl aller vom Backend innerhalb eines bestimmten Zeitraums bedienten Anforderungen, wenn das API-Caching aktiviert ist Die Statistik `Sum` repräsentiert diese Metrik, also die Gesamtzahl der Cache-Fehler innerhalb des angegebenen Zeitraums. Die Statistik `Average` repräsentiert die Cache-Fehlerrate, also die Gesamtzahl der Cache-Fehler geteilt durch die Gesamtzahl der Anforderungen innerhalb des Zeitraums. Der Nenner entspricht der Count-Metrik (unten). Unit: Count

Metrik	Beschreibung
Count	Die Gesamtzahl der API-Anforderungen innerhalb eines bestimmten Zeitraums Die Statistik `SampleCount` stellt diese Metrik dar. Unit: Count
IntegrationLatency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage an das Backend weiterleitet und zu dem Zeitpunkt, zu dem es eine Antwort vom Backend erhält. Unit: Millisecond
Latency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage von einem Client empfängt, und dem Zeitpunkt, an dem es eine Antwort an den Client zurückgibt. Die Latenzzeit schließt die Integrationslatenzzeit und anderen API Gateway-Overhead ein. Unit: Millisecond

Dimensionen für Metriken

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

> **Note**
>
> API Gateway entfernt Nicht-ASCII-Zeichen aus der ApiName Dimension, bevor Metriken an gesendet werden CloudWatch. Wenn der APIName keine ASCII-Zeichen enthält, wird die API ID als ApiName verwendet.

Dimension	Beschreibung
ApiName	Filtert API Gateway-Metriken für die REST-API mit dem angegebenen API-Namen.
ApiName, Method, Resource, Stage	Filtert API Gateway-Metriken für die API-Methode mit dem angegebenen API-Namen, der Stufe, Ressource und Methode. API Gateway sendet diese Metriken nur, wenn Sie die detaillierten CloudWatch Metriken explizit aktiviert haben. Wählen Sie in der Konsole eine Stufe aus, und klicken Sie dann unter Ptotokolle und Nachverfolgung auf Bearbeiten. Wählen Sie Detaillierte Metriken aus und klicken Sie dann auf Änderungen speichern. Alternativ können Sie den Befehl update-stage AWS CLI aufrufen, um die Eigenschaft `metricsEnabled` auf zu aktualisieren `true`. Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen Gebühren belastet. Preisinformationen finden Sie unter Amazon CloudWatch-Preise.
ApiName, Stage	Filtert API Gateway-Metriken für die API-Stufenressource mit dem angegebenen API-Namen und der Stufe.

CloudWatch Metriken mit dem API-Dashboard in API Gateway anzeigen

Sie können das API-Dashboard in der API Gateway Console verwenden, um die CloudWatch Metriken Ihrer bereitgestellten API in API Gateway anzuzeigen. Diese werden als eine Zusammenfassung der API-Aktivität im Zeitverlauf angezeigt.

Themen

- Voraussetzungen

- Untersuchen von API-Aktivitäten im Dashboard

Voraussetzungen

1. Sie müssen über eine in API Gateway erstellte API verfügen. Folgen Sie den Anweisungen in REST-API in API Gateway entwickeln.
2. Sie müssen die API mindestens einmal bereitgestellt haben. Folgen Sie den Anweisungen in Bereitstellen einer REST-API in Amazon API Gateway.

Untersuchen von API-Aktivitäten im Dashboard

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine API aus.
3. Wählen Sie im Haupt-Navigationsbereich Dashboard aus.
4. Wählen Sie für Stufe die gewünschte Stufe aus.
5. Wählen Sie Datumsbereich aus, um einen Datumsbereich anzugeben.
6. Aktualisieren Sie bei Bedarf einzelne Metriken in separaten Grafiken mit den Titeln API-Aufrufe, Latenz, Integrationslatenz, Latenz, 4xx-Fehler und 5xx-Fehler.

> **Tip**
>
> Um CloudWatch Metriken auf Methodenebene zu untersuchen, stellen Sie sicher, dass Sie CloudWatch Logs auf Methodenebene aktiviert haben. Weitere Informationen zum Einrichten der Protokollierung auf Methodenebene finden Sie unter Aktualisieren der Stufeneinstellungen mithilfe der API Gateway-Konsole.

API Gateway Gateway-Metriken in der CloudWatch Konsole anzeigen

Metriken werden zunächst nach dem Service-Namespace und anschließend nach den verschiedenen Dimensionskombinationen in den einzelnen Namespaces gruppiert. Um die Metriken für Ihre API auf Metrikebene anzuzeigen, aktivieren Sie detaillierte Metriken. Weitere Informationen finden Sie unter the section called “Aktualisieren von Stufeneinstellungen”.

So zeigen Sie API Gateway Gateway-Metriken mit der CloudWatch Konsole an

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Ändern Sie, falls erforderlich, die AWS-Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden.
3. Wählen Sie im Navigationsbereich Metrics aus.
4. Klicken Sie auf der Registerkarte All metrics auf API Gateway.
5. Klicken Sie auf den Bereich By Stage, um Metriken nach Phase anzuzeigen. Wählen Sie dann Ihre APIs und Metriknamen aus.
6. Klicken Sie auf den Bereich By Api Name, um Metriken für eine bestimmte API anzuzeigen. Wählen Sie dann Ihre APIs und Metriknamen aus.

So zeigen Sie Metriken mit der AWS CLI an

1. Geben Sie in der Befehlszeile den folgenden Befehl ein, um Metriken aufzulisten:

   ```
   aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
   ```

   Nachdem Sie eine Metrik erstellt haben, warten Sie bis zu 15 Minuten, bis die Metrik angezeigt wird. Verwenden Sie get-metric-dataoder, um Metrikstatistiken schneller zu sehen get-metric-statistics.
2. Rufen Sie den folgenden Befehl auf, um bestimmte Statistiken (z. B. `Average`) über einen Zeitraum von 5-Minuten-Intervallen anzuzeigen:

   ```
   aws cloudwatch get-metric-statistics --namespace AWS/ApiGateway --metric-name Count
    --start-time 2011-10-03T23:00:00Z --end-time 2017-10-05T23:00:00Z --period 300 --
   statistics Average
   ```

API Gateway Gateway-Protokollereignisse in der CloudWatch Konsole anzeigen

Voraussetzungen

1. Sie müssen über eine in API Gateway erstellte API verfügen. Folgen Sie den Anweisungen in REST-API in API Gateway entwickeln.

2. Sie müssen die API mindestens einmal in der API bereitgestellt haben. Befolgen Sie die Anweisungen unter Bereitstellen einer REST-API in Amazon API Gateway und REST-API im Amazon API Gateway aufrufen.
3. Sie müssen CloudWatch Logs für eine Phase aktiviert haben. Folgen Sie den Anweisungen in CloudWatch Protokollierung für eine REST-API in API Gateway einrichten.

Um protokollierte API-Anfragen und -Antworten mit der CloudWatch Konsole anzuzeigen

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Ändern Sie, falls erforderlich, die AWS-Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden. Weitere Informationen finden Sie unter - Regionen und Endpunkte.
3. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
4. Wählen Sie in der Tabelle Protokollgruppen eine Protokollgruppe mit dem Namen API-Gateway-Execution-Logs_ {}/{stage-name} aus. rest-api-id
5. Wählen Sie in der Tabelle Log Streams einen Protokoll-Stream aus. Sie können den gewünschten Protokoll-Stream anhand des Zeitstempels finden.
6. Klicken Sie auf Text, um den reinen Text anzuzeigen, oder klicken Sie auf Row, um das Ereignis zeilenweise anzuzeigen.

> **Important**
>
> CloudWatch ermöglicht das Löschen von Protokollgruppen oder Streams. Löschen Sie API Gateway-API-Protokollgruppen oder -Streams nicht manuell. Lassen Sie API Gateway diese Ressourcen verwalten. Wenn Sie Protokollgruppen oder Streams manuell löschen, werden API-Anforderungen und -Antworten möglicherweise nicht protokolliert. In diesem Fall können Sie die gesamte Protokollgruppe für die API löschen und die API erneut bereitstellen. Dies liegt daran, dass API Gateway Protokollgruppen oder -Streams einer API-Stufe zum Zeitpunkt der Bereitstellung erstellt.

Überwachungstools in AWS

AWS bietet verschiedene Tools, mit denen Sie API Gateway überwachen können. Sie können einige dieser Tools für die automatische Überwachung konfigurieren, während andere manuellen Eingriff erfordern. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Automatisierte Überwachungstools in AWS

Sie können die folgenden automatisierten Tools zur Überwachung von API Gateway verwenden und auftretende Probleme melden:

- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (Amazon SNS) - Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Status befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachung der REST-API-Ausführung mit CloudWatch Amazon-Metriken.
- Amazon CloudWatch Logs — Überwachen, speichern und greifen Sie auf Ihre Protokolldateien aus AWS CloudTrail oder anderen Quellen zu. Weitere Informationen finden Sie unter Was ist CloudWatch Logs? im CloudWatch Amazon-Benutzerhandbuch.
- Amazon EventBridge (früher CloudWatch Events genannt) — Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder Streams weiter, um Änderungen vorzunehmen, Statusinformationen zu erfassen und Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Benutzerhandbuch.
- AWS CloudTrail Protokollüberwachung — Teilen Sie Protokolldateien zwischen Konten, überwachen CloudTrail Sie Protokolldateien in Echtzeit, indem Sie sie an CloudWatch Logs senden, schreiben Sie Anwendungen zur Protokollverarbeitung in Java und stellen Sie sicher, dass sich Ihre Protokolldateien nach der Lieferung von nicht geändert haben CloudTrail. Weitere Informationen finden Sie unter Arbeiten mit CloudTrail Protokolldateien im AWS CloudTrail Benutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung von API Gateway besteht darin, die Elemente, die von den CloudWatch Alarmen nicht abgedeckt werden, manuell zu überwachen. Das API Gateway

und andere AWS Konsolen-Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung. CloudWatch Wir empfehlen, auch die Protokolldateien für die API-Ausführung zu überprüfen.

- Das API Gateway-Dashboard zeigt die folgenden Statistiken für eine bestimmte API-Stufe in einem bestimmten Zeitraum:
  - API Calls
  - Cache Hit, nur wenn das API-Caching aktiviert ist.
  - Cache Miss, nur wenn das API-Caching aktiviert ist.
  - Latency
  - Integration Latency
  - 4XX Error
  - 5XX Error
- Auf der CloudWatch Startseite wird Folgendes angezeigt:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können CloudWatch Sie Folgendes verwenden:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Suchen und durchsuchen Sie alle Ihre AWS Ressourcenmetriken
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

CloudWatch Alarme zur Überwachung des API Gateway erstellen

Sie können einen CloudWatch Alarm erstellen, der eine Amazon SNS SNS-Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum und führt eine oder mehrere Aktionen durch, die vom Wert der Metrik im Vergleich zu einem festgelegten Schwellenwert in einer Reihe von Zeiträumen abhängt. Die Aktion ist eine Benachrichtigung, die an ein Amazon SNS-Thema oder eine Auto Scaling-Richtlinie gesendet wird. Bei Alarmen werden nur Aktionen für anhaltende Statusänderungen ausgelöst. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der

Zustand muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein.

CloudWatch Protokollierung für eine REST-API in API Gateway einrichten

Um Probleme im Zusammenhang mit der Ausführung von Anfragen oder dem Client-Zugriff auf Ihre API zu beheben, können Sie Amazon CloudWatch Logs zur Protokollierung von API-Aufrufen aktivieren. Weitere Informationen zu finden Sie CloudWatch unterthe section called “CloudWatch Metriken”.

CloudWatch Protokollformate für API Gateway

Es gibt zwei Arten der API-Anmeldung CloudWatch: Ausführungsprotokollierung und Zugriffsprotokollierung. Bei der Ausführungsprotokollierung verwaltet API Gateway die CloudWatch Protokolle. Der Prozess umfasst das Erstellen von Protokollgruppen und -Streams sowie die Meldung der Anforderungen und Antworten jedes Aufrufers an Protokoll-Streams.

Zu den protokollierten Daten gehören Fehler oder Ausführungsspuren (wie Anforderungs- oder Antwortparameterwerte oder Payloads), von Lambda-Autorisierern (früher bekannt als benutzerdefinierte Autorisierer) verwendete Daten, ob API-Schlüssel erforderlich sind, ob Nutzungspläne aktiviert sind und andere Informationen. API Gateway redigiert Autorisierungsheader, API-Schlüsselwerte und ähnliche sensible Anforderungsparameter aus den protokollierten Daten.

Wenn Sie eine API bereitstellen, erstellt API Gateway eine Protokollgruppe und -Streams unter der Protokollgruppe. Die Protokollgruppe wird nach dem `API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`-Format benannt. In jeder Protokollgruppe, werden die Protokolle weiter in Protokoll-Streams unterteilt, die bei Meldung von Protokolldaten nach Last Event Time sortiert werden.

In der Zugriffsprotokollierung protokollieren Sie, als API-Entwickler, wer auf Ihre API zugegriffen hat und wie der Aufrufer auf die API zugegriffen hat. Erstellen Sie eine eigene Protokollgruppe oder wählen Sie eine vorhandene Protokollgruppe aus, die von API Gateway verwaltet werden kann. Um die Zugriffsdetails anzugeben, wählen Sie `$context`Variablen, ein Protokollformat und ein Protokollgruppenziel aus.

Das Zugriffsprotokoll-Format muss mindestens `$context.requestId` oder `$context.extendedRequestId` enthalten. Es hat sich bewährt, `$context.requestId` und `$context.extendedRequestId` in Ihr Protokollformat aufzunehmen.

`$context.requestId`

Dadurch wird der Wert in der `x-amzn-RequestId` Kopfzeile protokolliert. Clients können den Wert im `x-amzn-RequestId`-Header durch einen Wert im Format einer UUID (Universally Unique Identifier) überschreiben. API Gateway gibt diese Anforderungs-ID im `x-amzn-RequestId`-Antwort-Header zurück. API Gateway ersetzt überschriebene Anforderungs-IDs, die nicht das Format einer UUID haben, durch *`UUID`*`_REPLACED_INVALID_REQUEST_ID` in Ihren Zugriffsprotokollen.

`$context.extendedRequestId`

Die ExtendedRequestID ist eine eindeutige ID, die API Gateway generiert. API Gateway gibt diese Anforderungs-ID im `x-amz-apigw-id`-Antwort-Header zurück. Ein API-Aufrufer kann diese Anforderungs-ID nicht bereitstellen oder überschreiben. Möglicherweise müssen Sie diesen Wert dem AWS Support zur Verfügung stellen, um die Fehlerbehebung für Ihre API zu unterstützen. Weitere Informationen finden Sie unter the section called “`$context`Variablen für Datenmodelle, Autorisierer, Zuordnungsvorlagen und Zugriffsprotokollierung CloudWatch ”.

> **Note**
>
> Es werden nur `$context` Variablen unterstützt.

Wählen Sie ein Protokollformat aus, das auch von Ihrem analytischen Backend genutzt wird, z. B. Common Log Format (CLF), JSON, XML, oder CSV. Anschließend können Sie die Zugriffsprotokolle direkt eingeben und Ihre Metriken berechnen und rendern lassen. Um das Protokollformat zu definieren, legen Sie den ARN der Protokollgruppe in der Eigenschaft accessLogSettings/DestinationArn auf der Bühne fest. Sie können einen Protokollgruppen-ARN in der CloudWatch Konsole abrufen. Um das Format des Zugriffsprotokolls zu definieren, legen Sie in der Eigenschaft accessLogSetting/format auf der Bühne ein ausgewähltes Format fest.

Beispiele für einige häufig verwendete Zugriffsprotokollformate werden in der API Gateway-Konsole dargestellt und wie folgt aufgeführt.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller $context.identity.user
 [$context.requestTime]"$context.httpMethod $context.resourcePath
```

```
 $context.protocol" $context.status $context.responseLength $context.requestId
 $context.extendedRequestId
```

- JSON:

```
{ "requestId":"$context.requestId",
 "extendedRequestId":"$context.extendedRequestId","ip": "$context.identity.sourceIp",
 "caller":"$context.identity.caller", "user":"$context.identity.user",
 "requestTime":"$context.requestTime", "httpMethod":"$context.httpMethod",
 "resourcePath":"$context.resourcePath", "status":"$context.status",
 "protocol":"$context.protocol", "responseLength":"$context.responseLength" }
```

- XML:

```
<request id="$context.requestId"> <extendedRequestId>$context.extendedRequestId</
extendedRequestId> <ip>$context.identity.sourceIp</ip> <caller>
$context.identity.caller</caller> <user>$context.identity.user</user> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <resourcePath>$context.resourcePath</resourcePath> <status>$context.status</status>
 <protocol>$context.protocol</protocol> <responseLength>$context.responseLength</
responseLength> </request>
```

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Berechtigungen für die Protokollierung CloudWatch

Um CloudWatch Logs zu aktivieren, müssen Sie API Gateway die Erlaubnis erteilen, Logs CloudWatch für Ihr Konto zu lesen und zu schreiben. Die `AmazonAPIGatewayPushToCloudWatchLogs` verwaltete Richtlinie (mit dem ARN `arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs`) verfügt über alle erforderlichen Berechtigungen:

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

```
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> API Gateway ruft AWS Security Token Service auf, um die IAM-Rolle zu übernehmen. Stellen Sie also sicher, dass diese für die Region aktiviert AWS STS ist. Weitere Informationen finden Sie unter AWS STS in einer AWS Region verwalten.

Um Ihrem Konto diese Berechtigungen zu gewähren, erstellen Sie eine IAM-Rolle `apigateway.amazonaws.com` als vertrauenswürdige Entität, fügen Sie der IAM-Rolle die obige Richtlinie hinzu und legen Sie den ARN der IAM-Rolle in der cloudWatchRole Eigenschaft Arn in Ihrem Konto fest. Sie müssen die Eigenschaft cloudWatchRoleArn für jede AWS Region, in der Sie Logs aktivieren möchten, separat festlegen. CloudWatch

Wenn Sie beim Einstellen des ARN für die IAM-Rolle eine Fehlermeldung erhalten, überprüfen Sie Ihre AWS Security Token Service Kontoeinstellungen, um sicherzustellen, dass diese in der Region, die Sie verwenden, aktiviert AWS STS ist. Weitere Informationen zur Aktivierung AWS STS finden Sie im IAM-Benutzerhandbuch unter AWS STS in einer AWS Region verwalten.

CloudWatch API-Protokollierung mit der API Gateway Gateway-Konsole einrichten

Um die CloudWatch API-Protokollierung einzurichten, müssen Sie die API in einer Phase bereitgestellt haben. Sie müssen auch einen entsprechenden CloudWatch Logs-Rollen-ARN für Ihr Konto konfiguriert haben.

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Wählen Sie im Hauptnavigationsbereich Settings (Einstellungen) und dann unter Logging (Protokollierung) die Option Edit (Bearbeiten) aus.
3. Geben Sie für die CloudWatch Protokollrolle ARN einen ARN einer IAM-Rolle mit den entsprechenden Berechtigungen ein. Sie müssen dies einmal für jeden tun AWS-Konto , der APIs mithilfe von API Gateway erstellt.
4. Klicken Sie im Hauptnavigationsbereich auf APIs und führen Sie eine der folgenden Aktionen aus:

   a. Wählen Sie eine vorhandene API und anschließend eine Stufe aus.

   b. Erstellen Sie eine API und stellen Sie diese dann einer Stufe bereit.
5. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
6. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
7. So aktivieren Sie die Ausführungsprotokollierung:

   a. Wählen Sie im Dropdownmenü Protokolle eine CloudWatch Protokollierungsebene aus. Die Protokollierungsebenen sind die folgenden:

      - Aus — Die Protokollierung ist für diese Phase nicht aktiviert.
      - Nur Fehler — Die Protokollierung ist nur für Fehler aktiviert.
      - Fehler und Informationsprotokolle — Die Protokollierung ist für alle Ereignisse aktiviert.
      - Vollständige Anfrage- und Antwortprotokolle — Die detaillierte Protokollierung ist für alle Ereignisse aktiviert. Dies kann zur Fehlerbehebung bei APIs hilfreich sein, kann aber dazu führen, dass sensible Daten protokolliert werden.

        > **Note**
        >
        > Es wird empfohlen, Full Request and Response Logs (Vollständige Anforderungs- und Antwortprotokolle) nicht zu verwenden.

   b. Falls gewünscht, wählen Sie Detaillierte Metriken aus, um detaillierte CloudWatch Metriken zu aktivieren.

   Weitere Informationen zu CloudWatch Metriken finden Sie unterthe section called “CloudWatch Metriken”.
8. So aktivieren Sie die Zugriffsprotokollierung:

a. Aktivieren Sie die Option Custom access logging (Benutzerdefinierte Zugriffsprotokollierung).

b. Geben Sie den ARN einer Protokollgruppe in Access Log Destination ARN (Ziel-ARN des Zugriffsprotokolls) ein. Das ARN-Format ist `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.

c. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Sie können zwischen CLF, JSON, XML oder CSV wählen. Weitere Informationen zu Beispielprotokollformaten finden Sie unter the section called “CloudWatch Protokollformate für API Gateway”.

9. Wählen Sie Änderungen speichern aus.

> **Note**
>
> Sie können die Ausführungs- und Zugriffsprotokollierung unabhängig voneinander aktivieren.

API Gateway kann nun Anforderungen an Ihre API protokollieren. Sie müssen die API nicht erneut bereitstellen, wenn Sie die Stufeneinstellungen, Protokolle oder Stufenvariablen aktualisieren.

Richten Sie die CloudWatch API-Protokollierung ein mit AWS CloudFormation

Verwenden Sie die folgende AWS CloudFormation Beispielvorlage, um eine Amazon CloudWatch Logs-Protokollgruppe zu erstellen und die Ausführungs- und Zugriffsprotokollierung für eine Phase zu konfigurieren. Um CloudWatch Logs zu aktivieren, müssen Sie API Gateway die Erlaubnis erteilen, Logs CloudWatch für Ihr Konto zu lesen und zu schreiben. Weitere Informationen finden Sie unter Konto mit IAM-Rolle verknüpfen im AWS CloudFormation -Benutzerhandbuch.

```
TestStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: test
    RestApiId: !Ref MyAPI
    DeploymentId: !Ref Deployment
    Description: "test stage description"
    MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        LoggingLevel: INFO
    AccessLogSetting:
```

```
        DestinationArn: !GetAtt MyLogGroup.Arn
        Format: $context.extendedRequestId $context.identity.sourceIp
$context.identity.caller $context.identity.user [$context.requestTime]
"$context.httpMethod $context.resourcePath $context.protocol" $context.status
$context.responseLength $context.requestId
 MyLogGroup:
   Type: AWS::Logs::LogGroup
   Properties:
     LogGroupName: !Join
       - '-'
       - - !Ref MyAPI
         - access-logs
```

Protokollieren von API-Aufrufen bei Amazon Data Firehose

Um Probleme im Zusammenhang mit dem Client-Zugriff auf Ihre API zu beheben, können Sie API-Aufrufe bei Amazon Data Firehose protokollieren. Weitere Informationen zu Firehose finden Sie unter Was ist Amazon Data Firehose? .

Für die Zugriffsprotokollierung können Sie nur CloudWatch oder Firehose aktivieren — Sie können nicht beide aktivieren. Sie können jedoch die Ausführungsprotokollierung und Firehose für die Zugriffsprotokollierung aktivieren CloudWatch .

Themen

- Firehose-Protokollformate für API Gateway
- Berechtigungen für die Firehose-Protokollierung
- Richten Sie die Firehose-Zugriffsprotokollierung mithilfe der API Gateway Gateway-Konsole ein

Firehose-Protokollformate für API Gateway

Die Firehose-Protokollierung verwendet dasselbe Format wie die CloudWatch Protokollierung.

Berechtigungen für die Firehose-Protokollierung

Wenn die Firehose-Zugriffsprotokollierung auf einer Stufe aktiviert ist, erstellt API Gateway eine dienstverknüpfte Rolle in Ihrem Konto, sofern die Rolle noch nicht vorhanden ist. Die Rolle wird `AWSServiceRoleForAPIGateway` benannt und die verwaltete `APIGatewayServiceRolePolicy`-Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter Verwenden serviceverknüpfter Rollen.

> **Note**
>
> Der Name Ihres Firehose-Streams muss sein. `amazon-apigateway-{your-stream-name}`

Richten Sie die Firehose-Zugriffsprotokollierung mithilfe der API Gateway Gateway-Konsole ein

Zum Einrichten der API-Protokollierung müssen Sie die API für eine Stufe bereitgestellt haben. Sie müssen auch einen Firehose-Stream erstellt haben.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Führen Sie eine der folgenden Aktionen aus:
   a. Wählen Sie eine vorhandene API und anschließend eine Stufe aus.
   b. Erstellen Sie eine API und stellen Sie diese einer Stufe bereit.
3. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
4. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
5. So aktivieren Sie die Zugriffsprotokollierung für einen Firehose-Stream:
   a. Aktivieren Sie die Option Custom access logging (Benutzerdefinierte Zugriffsprotokollierung).
   b. Geben Sie für Access Log Destination ARN den ARN eines Firehose ein. Das ARN-Format ist `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

      > **Note**
      >
      > Der Name Ihres Firehose-Streams muss sein. `amazon-apigateway-{your-stream-name}`

   c. Geben Sie unter Log Format (Protokollformat) ein Protokollformat ein. Sie können zwischen CLF, JSON, XML oder CSV wählen. Weitere Informationen zu Beispielprotokollformaten finden Sie unter the section called “CloudWatch Protokollformate für API Gateway”.

6. Wählen Sie Änderungen speichern aus.

API Gateway ist jetzt bereit, Anfragen an Ihre API an Firehose zu protokollieren. Sie müssen die API nicht erneut bereitstellen, wenn Sie die Stufeneinstellungen, Protokolle oder Stufenvariablen aktualisieren.

Ablaufverfolgung von Benutzeranforderungen an REST-APIs mithilfe von X-Ray

Sie können AWS X-Ray verwenden, um Benutzeranforderungen auf ihrem Weg durch Ihre Amazon-API-Gateway-REST-APIs zu den zugrunde liegenden Diensten zu verfolgen und zu analysieren. API Gateway unterstützt die X-Ray-Ablaufverfolgung für alle API Gateway-REST-API-Endpunkttypen: Regional, Edge-optimiert und privat. Sie können X-Ray mit Amazon API Gateway in allen AWS Regionen verwenden, in denen X-Ray verfügbar ist.

Da X-Ray Ihnen einen end-to-end Überblick über eine gesamte Anfrage bietet, können Sie Latenzen in Ihren APIs und deren Backend-Services analysieren. Sie können eine X-Ray-Servicezuordnung verwenden, um die Latenz einer gesamten Anforderung und der der nachgeschalteten Services anzuzeigen, die in X-Ray integriert sind. Sie können auch Samplingregeln konfigurieren, um X-Ray mitzuteilen, welche Anforderungen mit welchen Abtastraten gemäß den von Ihnen angegebenen Kriterien aufgezeichnet werden sollen.

Wenn Sie eine API Gateway-API von einem Service aufrufen, der bereits verfolgt wird, wird API Gateway die Ablaufverfolgung auch dann ausführen, wenn die X-Ray-Ablaufverfolgung in der API nicht aktiviert ist.

Sie können X-Ray für eine API-Stufe aktivieren, indem Sie die API Gateway-Konsole oder die API Gateway-API oder -CLI verwenden.

Themen

- Einrichtung AWS X-Ray mit API-Gateway-REST-APIs
- Verwenden von AWS X-Ray Service Maps und Trace Views mit API Gateway
- Konfiguration von AWS X-Ray Sampling-Regeln für API Gateway Gateway-APIs
- AWS X-Ray Traces für Amazon API Gateway Gateway-APIs verstehen

Einrichtung AWS X-Ray mit API-Gateway-REST-APIs

In diesem Abschnitt finden Sie detaillierte Informationen zum Einrichten von AWS X-Ray mit API-Gateway-REST-APIs.

Themen

X-Ray-Ablaufverfolgungsmodi für API Gateway

Der Pfad einer Anforderung über Ihre Anwendung wird mit einer Nachverfolgungs-ID verfolgt. Eine Ablaufverfolgung erfasst alle von einer einzelnen Anforderung generierten Segmente; in der Regel handelt es sich dabei um eine HTTP `GET`- oder `POST`-Anforderung.

Es gibt zwei Ablaufverfolgungsmodi für eine API Gateway-API:

- Passiv: Dies ist die Standardeinstellung, wenn Sie die X-Ray-Ablaufverfolgung in einer API-Stufe nicht aktiviert haben. Bei diesem Ansatz wird die API Gateway-API nur nachverfolgt, wenn X-Ray für einen Upstream-Service aktiviert wurde.
- Aktiv: Wenn eine API Gateway-API-Stufe diese Einstellung aufweist, stellt API Gateway die API-Aufrufanfragen automatisch auf Basis des von X-Ray bereitgestellten Sampling-Algorithmus zusammen.

  Wenn die aktive Ablaufverfolgung in einer Stufe aktiviert ist, wird API Gateway in Ihrem Konto eine servicebezogene Rolle erstellen, sofern die Rolle nicht bereits vorhanden ist. Die Rolle wird benannt `AWSServiceRoleForAPIGateway`und die `APIGatewayServiceRolePolicy` verwaltete Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter Verwenden serviceverknüpfter Rollen.

  > **Note**
  >
  > X-Ray wendet einen Sampling-Algorithmus an, um eine effizient Ablaufverfolgung zu gewährleisten und gleichzeitig ein repräsentatives Beispiel für die Anforderungen bereitzustellen, die von Ihrer API bedient werden. Der Standardeinstellung für den

Sampling-Algorithmus lautet 1 Anforderung pro Sekunde, wobei 5 Prozent der Anforderungen, die über dieses Limit hinausgehen, auch erfasst werden.

Sie können den Ablaufverfolgungsmodus für Ihre API ändern, indem Sie die API Gateway Gateway-Verwaltungskonsole, die API Gateway Gateway-CLI oder ein AWS SDK verwenden.

Berechtigungen für die X-Ray-Ablaufverfolgung

Wenn Sie die X-Ray-Ablaufverfolgung in einer Stufe aktivieren, wird API Gateway in Ihrem Konto eine servicebezogene Rolle erstellen, sofern die Rolle nicht bereits vorhanden ist. Die Rolle wird benannt `AWSServiceRoleForAPIGateway`und die `APIGatewayServiceRolePolicy` verwaltete Richtlinie wird an sie angehängt. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter Verwenden serviceverknüpfter Rollen.

Aktivieren der X-Ray-Ablaufverfolgung in der API Gateway-Konsole

Sie können die Amazon API Gateway-Konsole verwenden, um die aktive Ablaufverfolgung in einer API-Stufe zu aktivieren.

Diese Schritte gehen davon aus, dass Sie die API bereits in einer Stufe bereitgestellt haben.

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie Ihre API und dann im Hauptnavigationsbereich die Option Stages (Stufe).
3. Wählen Sie im Bereich Stages (Stufen) eine Stufe aus.
4. Wählen Sie im Abschnitt Logs and tracing (Protokolle und Nachverfolgung) die Option Edit (Bearbeiten) aus.
5. Sie können die aktive X-Ray-Nachverfolgung aktivieren, indem Sie X-Ray tracing (X-Ray-Nachverfolgung) auswählen, um die X-Ray-Nachverfolgung zu aktivieren.
6. Wählen Sie Änderungen speichern aus.

Nachdem Sie X-Ray für Ihre API-Stufe aktiviert haben, können Sie die Ablaufverfolgungen und Service-Übersichten mithilfe der X-Ray-Verwaltungskonsole anzeigen.

Aktivieren der AWS X-Ray Ablaufverfolgung mit der API Gateway Gateway-CLI

AWS CLI Um die aktive Röntgenverfolgung für eine API-Stufe zu aktivieren, wenn Sie die Phase erstellen, rufen Sie den `create-stage`Befehl wie im folgenden Beispiel auf:

```
aws apigateway create-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=true
```

Es folgt eine Beispielausgabe für einen erfolgreichen Aufruf:

```
{
    "tracingEnabled": true,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

AWS CLI Um das aktive X-Ray-Tracing für eine API-Stufe zu deaktivieren, wenn Sie die Phase erstellen, rufen Sie den `create-stage`Befehl wie im folgenden Beispiel auf:

```
aws apigateway create-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=false
```

Es folgt eine Beispielausgabe für einen erfolgreichen Aufruf:

```
{
    "tracingEnabled": false,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
```

```
}
```

AWS CLI Um das aktive X-Ray-Tracing für eine API zu aktivieren, die bereits bereitgestellt wurde, rufen Sie den `update-stage`Befehl wie folgt auf:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --patch-operations op=replace,path=/tracingEnabled,value=true
```

AWS CLI Um das aktive X-Ray-Tracing für eine API zu deaktivieren, die bereits bereitgestellt wurde, rufen Sie den `update-stage`Befehl wie im folgenden Beispiel auf:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --region {region} \
    --patch-operations op=replace,path=/tracingEnabled,value=false
```

Es folgt eine Beispielausgabe für einen erfolgreichen Aufruf:

```
{
    "tracingEnabled": false,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533850033,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Sobald Sie X-Ray für Ihre API-Stufe aktiviert haben, verwenden Sie die X-Ray-CLI zum Abrufen von Ablaufverfolgungsinformationen. Weitere Informationen finden Sie unter Verwenden der AWS X-Ray API mit der AWS CLI.

Verwenden von AWS X-Ray Service Maps und Trace Views mit API Gateway

In diesem Abschnitt finden Sie detaillierte Informationen zur Verwendung von AWS X-Ray-Service-Übersichten und Nachverfolgungsansichten mit API Gateway.

Ausführliche Informationen zu Service-Übersichten und Ablaufverfolgungsansichten und wie Sie diese interpretieren finden Sie unter AWS X-Ray -Konsole.

Themen

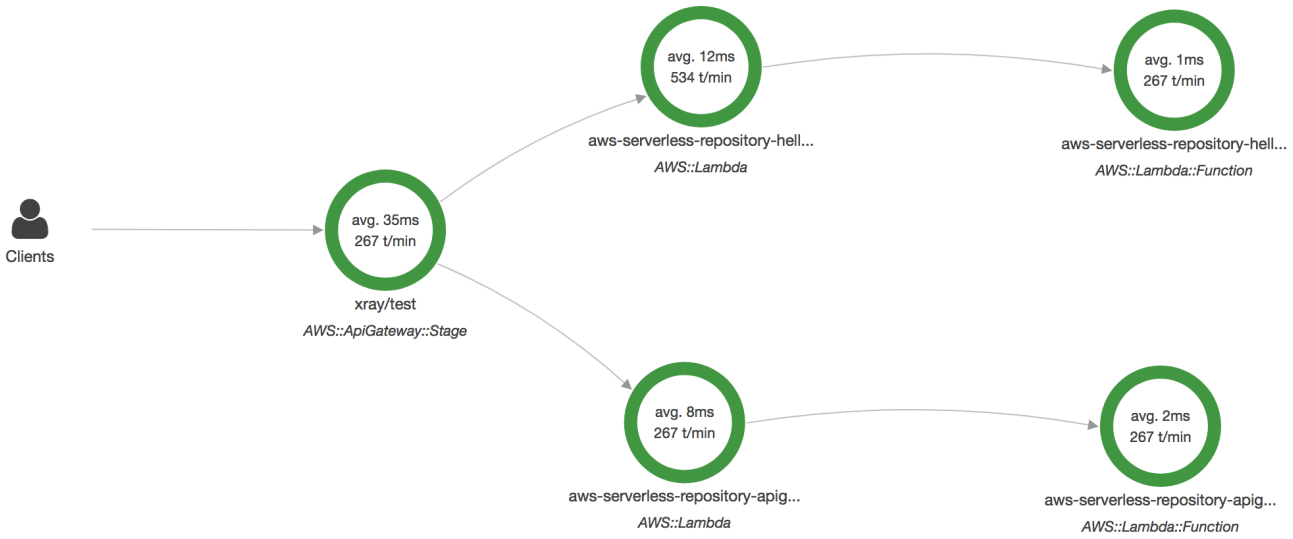
- Beispiel für eine X-Ray-Service-Übersicht
- Beispiel für eine X-Ray-Ablaufverfolgungsansicht

Beispiel für eine X-Ray-Service-Übersicht

AWS X-Ray Service Maps zeigen Informationen zu Ihrer API und all ihren nachgelagerten Diensten. Wenn X-Ray für eine API-Stufe in API Gateway aktiviert ist, wird in der Service-Übersicht ein Knoten angezeigt, der Informationen zur Gesamtzeit enthält, die im API Gateway-Service verbracht wurde. Sie erhalten detaillierte Informationen über den Antwortstatus und ein Histogramm der API-Reaktionszeit für den ausgewählten Zeitraum. Bei APIs, die in AWS Dienste wie AWS Lambda Amazon DynamoDB integriert sind, werden Sie mehr Knoten sehen, die Leistungskennzahlen für diese Dienste bereitstellen. Es wird eine Service-Übersicht für jede API-Stufe geben.

Das folgende Beispiel zeigt eine Service-Übersicht für die `test` Stufe einer aufgerufenen API `xray`. Diese API verfügt über eine Lambda-Integration mit einer Lambda-Genehmiger-Funktion und einer Lambda-Backend-Funktion. Die Knoten stellen den API Gateway-Service, den Lambda-Service und die beiden Lambda-Funktionen dar.

Eine detaillierte Erläuterung der Service-Übersichtsstruktur finden Sie unter Anzeigen der Service-Übersicht.



Sie können die Service-Übersicht vergrößern und eine Ansicht der Nachverfolgung für die API-Stufe anzeigen. Die Nachverfolgung zeigt detaillierte Informationen zu Ihrer API an, die als Segmente und Untersegmente dargestellt werden. Beispiel: Die Ablaufverfolgung für die Service-Übersicht umfasst die oben gezeigten Segmente für den Lambda-Service und die Lambda-Funktion. Weitere Informationen finden Sie unter AWS Lambda und. AWS X-Ray

Wenn Sie einen Knoten oder ein Edge in einer X-Ray-Service-Übersicht auswählen, zeigt die X-Ray-Konsole ein Histogramm der Latenzverteilung. Sie können ein Latenzhistogramm verwenden, um zu sehen, wie lange es dauert, bis ein Service seine Anforderungen erfüllt. Es folgt ein Histogramm der API Gateway-Stufe, die in der vorherigen Service-Übersicht `xray/test` benannt wurde.
Eine detaillierte Erklärung der Latenz-Verteilungshistogramme finden Sie unter Verwenden von Latenzhistogrammen in der AWS X-Ray -Konsole.

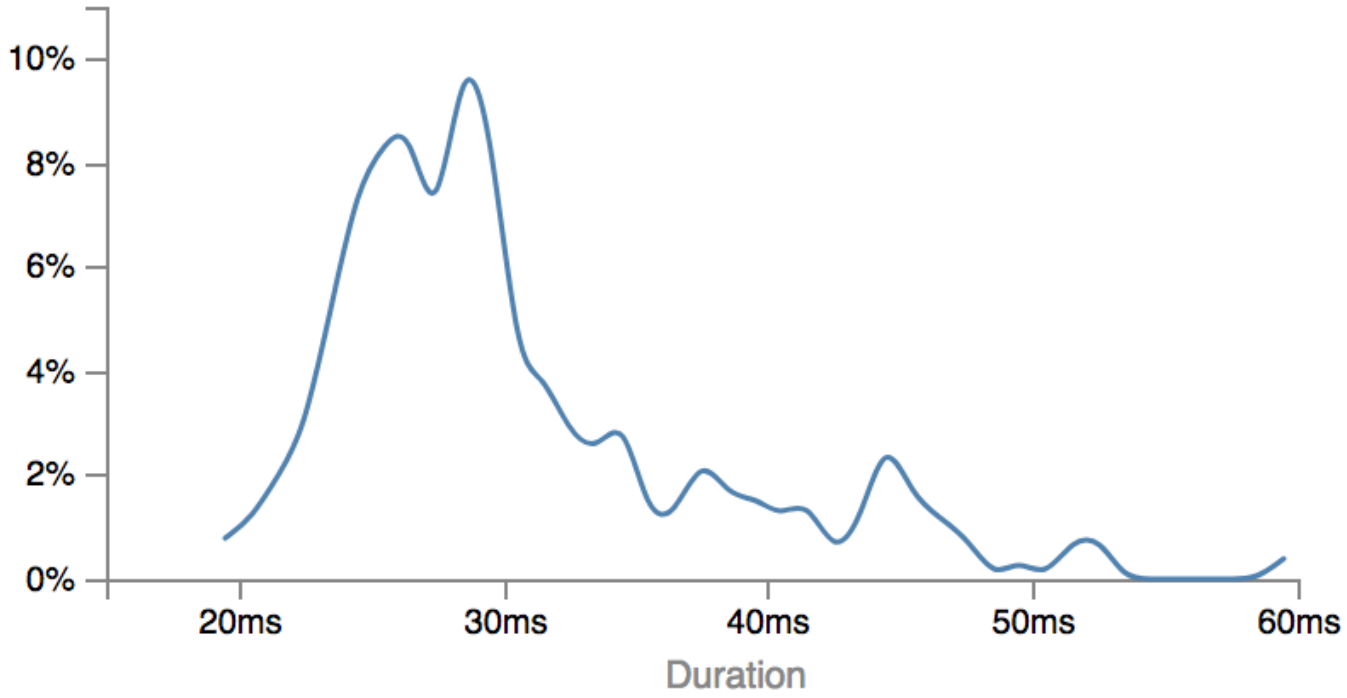
Service details

Name: xray/test

Type: AWS::ApiGateway::Stage

Response distribution

Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.



Response status

Choose response statuses to add to the filter when viewing traces.

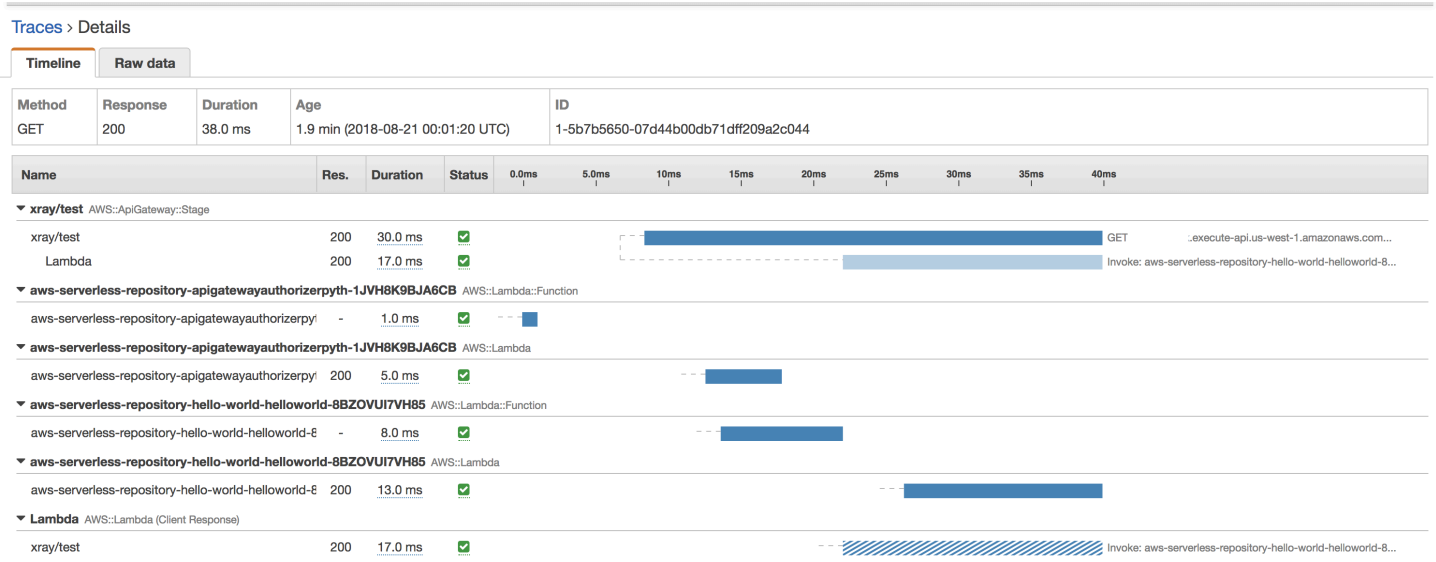
OK: 100% Error: 0%

Fault: 0% Throttle: 0%

Beispiel für eine X-Ray-Ablaufverfolgungsansicht

Das folgende Diagramm zeigt eine Ablaufverfolgungsansicht, die für die oben beschriebene Beispiel-API mit einer Lambda-Backend-Funktion und einer Lambda-Genehmiger-Funktion generiert wurde. Eine erfolgreiche API-Methodenanforderung wird mit einem Antwortcode von 200 angezeigt.

Eine detaillierte Erläuterung der Ablaufverfolgungsansichten finden Sie unter Anzeigen von Ablaufverfolgungen.



Konfiguration von AWS X-Ray Sampling-Regeln für API Gateway Gateway-APIs

Sie können die AWS X-Ray Konsole oder das SDK verwenden, um Sampling-Regeln für Ihre Amazon API Gateway zu konfigurieren. Eine Samplingregel gibt an, welche Anforderungen X-Ray für Ihre API aufzeichnen soll. Durch das Anpassen von Samplingregeln können Sie die Menge der von Ihnen aufgezeichneten Daten steuern und das Stichprobenverhalten im laufenden Betrieb ändern, ohne Ihren Code ändern oder neu implementieren zu müssen.

Bevor Sie Ihre X-Ray-Samplingregeln festlegen, lesen Sie die folgenden Themen im X-Ray-Entwicklerhandbuch:

- Sampling-Regeln in der AWS X-Ray Konsole konfigurieren
- Verwenden von Samplingregeln mit der X-Ray-API

Themen

- Optionswerte der X-Ray-Samplingregeln für API Gateway-APIs

- Beispiele für X-Ray-Samplingregeln

Optionswerte der X-Ray-Samplingregeln für API Gateway-APIs

Die folgenden X-Ray-Samplingoptionen sind für API Gateway relevant. String-Werte können Platzhalter verwenden, um ein einzelnes Zeichen (?) Oder null oder mehr Zeichen (*) zu finden. Weitere Informationen, einschließlich einer ausführlichen Erläuterung der Verwendung der Einstellungen für Reservoir und Rate, finden Sie unter Konfiguration der Probenahmeregeln in der AWS X-Ray Konsole.

- Regelname (Zeichenfolge) – Ein eindeutiger Name für die Regel.
- Priorität (Ganzzahl zwischen 1 und 9999) – Die Priorität der Samplingregel. Services werten Regeln in aufsteigender Reihenfolge der Priorität aus und treffen eine Sampleentscheidung mit der ersten übereinstimmenden Regel.
- Reservoir (nicht negative Ganzzahl) – Eine feste Anzahl übereinstimmender Anfragen an das Gerät pro Sekunde vor Anwendung des festen Satzes. Das Reservoir wird nicht direkt von Services verwendet, sondern gilt für alle Services, die die Regel gemeinsam verwenden.
- Rate (Anzahl zwischen 0 und 100) – Der Prozentsatz der übereinstimmenden Anfragen an das Gerät, nachdem das Reservoir erschöpft ist.
- Servicename (Zeichenfolge) – Der API-Stufenname in Form von ***{api-name}/{stage-name}***. Wenn Sie die PetStoreBeispiel-API beispielsweise in einer Phase mit dem Namen bereitstellen würden, würde der Wert für den Servicenamen **`test`**, den Sie in Ihrer Stichprobenregel angeben müssen, lauten**`pets/test`**.
- Servicetyp (Zeichenfolge) – Für eine API Gateway-API kann entweder **`AWS::ApiGateway::Stage`** oder **`AWS::ApiGateway::*`** angegeben werden.
- Host (Zeichenfolge) – Der Hostname aus dem HTTP-Host-Header. Setzen Sie dies auf **`*`**, um mit allen Hostnamen übereinzustimmen. Oder geben Sie einen vollständigen oder teilweisen Hostnamen an, z. B. **`api.example.com`** oder **`*.example.com`**.
- Ressourcen-ARN (Zeichenfolge) – Der ARN der API-Stufe, z. B. **`arn:aws:apigateway:`*`region`*`::/restapis/`*`api-id`*`/stages/`*`stage-name`***.

  Der Stufenname kann von der Konsole oder der API Gateway-CLI oder -API abgerufen werden. Weitere Informationen zu ARN-Formaten finden Sie unter Allgemeine Amazon Web Services-Referenz.
- HTTP-Methode (Zeichenfolge) – Die Methode, die gesampelt werden soll, z. B. **`GET`**.

- URL path (URL-Pfad) (Zeichenfolge) – Der URL-Pfad der Anforderung.
- (optional) Attribute (Schlüssel und Wert) – Header aus der ursprünglichen HTTP-Anforderung, z. B. **`Connection`**, **`Content-Length`** oder **`Content-Type`**. Jeder Attributwert kann bis zu 32 Zeichen lang sein.

Beispiele für X-Ray-Samplingregeln

Beispiel #1 für Samplingregeln

Diese Regel sampelt alle `GET` Anforderungen für die `testxray` API- `test` Stufe.

- Rule name (Regelname — **`test-sampling`**
- Priorität — **`17`**
- Reservoirgröße — **`10`**
- Bestimmtes Zeitintervall — **`10`**
- Servicename — **`testxray/test`**
- Servicetyp – **`AWS::ApiGateway::Stage`**
- HTTP-Methode — **`GET`**
- Ressourcen-ARN – **`*`**
- Host — **`*`**

Beispiel #2 für Samplingregeln

Diese Regel sampelt alle Anforderungen für die `testxray` API in der `prod`-Stufe.

- Rule name (Regelname — **`prod-sampling`**
- Priorität — **`478`**
- Reservoirgröße — **`1`**
- Bestimmtes Zeitintervall — **`60`**
- Servicename — **`testxray/prod`**
- Servicetyp – **`AWS::ApiGateway::Stage`**
- HTTP-Methode — **`*`**
- Ressourcen-ARN – **`*`**
- Host — **`*`**

- Attribute — **{}**

AWS X-Ray Traces für Amazon API Gateway Gateway-APIs verstehen

In diesem Abschnitt werden AWS X-Ray Trace-Segmente, Untersegmente und andere Trace-Felder für Amazon API Gateway Gateway-APIs behandelt.

Bevor Sie diesen Abschnitt lesen, lesen Sie sich die folgenden Themen im X-Ray-Entwicklerhandbuch durch:

- AWS X-Ray Konsole
- AWS X-Ray Dokumente segmentieren
- X-Ray-Konzepte

Themen

- Beispiele für Ablaufverfolgungsobjekte für eine API Gateway-API
- Verstehen der Ablaufverfolgung

Beispiele für Ablaufverfolgungsobjekte für eine API Gateway-API

In diesem Abschnitt werden einige der Objekte erläutert, die in einer Ablaufverfolgung für eine API Gateway-API angezeigt werden.

Anmerkungen

Anmerkungen können in Segmenten und Untersegmenten vorkommen. Sie werden als Filterausdrücke in Stichprobenregeln zum Filtern von Nachverfolgungs verwendet. Weitere Informationen finden Sie unter Konfigurieren von Samplingregeln in der AWS X-Ray -Konsole.

Im Folgenden sehen Sie ein Beispiel für ein `annotations`-Objekt, in dem eine API-Stufe durch die API-ID und den API-Stufennamen identifiziert wird:

```
"annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
}
```

AWS Ressourcendaten

Das `aws`-Objekt wird nur in Segmenten angezeigt. Im Folgenden finden Sie ein Beispiel für ein `aws` Objekt, das der Standard-Sampling-Regel entspricht. Eine ausführliche Erläuterung der Samplingregeln finden Sie unter Konfigurieren von Samplingregeln in der AWS X-Ray -Konsole.

```
"aws": {
    "xray": {
        "sampling_rule_name": "Default"
    },
    "api_gateway": {
        "account_id": "123412341234",
        "rest_api_id": "a1b2c3d4e5",
        "stage": "dev",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
}
```

Verstehen der Ablaufverfolgung

Im Folgenden finden Sie ein Ablaufverfolgungssegment für eine API Gateway-Stufe. Eine ausführliche Erläuterung der Felder, aus denen das Trace-Segment besteht, finden Sie im AWS X-Ray Entwicklerhandbuch unter AWS X-Ray Segmentdokumente.

```
        {
            "Document": {
                "id": "a1b2c3d4a1b2c3d4",
                "name": "testxray/dev",
                "start_time": 1533928226.229,
                "end_time": 1533928226.614,
                "metadata": {
                    "default": {
                        "extended_request_id": "abcde12345abcde=",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "http": {
                    "request": {
                        "url": "https://example.com/dev?
username=demo&message=hellofromdemo/",
                        "method": "GET",
                        "client_ip": "192.0.2.0",
                        "x_forwarded_for": true
                    },
```

```
                    "response": {
                        "status": 200,
                        "content_length": 0
                    }
                },
                "aws": {
                    "xray": {
                        "sampling_rule_name": "Default"
                    },
                    "api_gateway": {
                        "account_id": "123412341234",
                        "rest_api_id": "a1b2c3d4e5",
                        "stage": "dev",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "annotations": {
                    "aws:api_id": "a1b2c3d4e5",
                    "aws:api_stage": "dev"
                },
                "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
                "origin": "AWS::ApiGateway::Stage",
                "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/
stages/dev",
                "subsegments": [
                    {
                        "id": "abcdefgh12345678",
                        "name": "Lambda",
                        "start_time": 1533928226.233,
                        "end_time": 1533928226.6130002,
                        "http": {
                            "request": {
                                "url": "https://example.com/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
                                "method": "GET"
                            },
                            "response": {
                                "status": 200,
                                "content_length": 62
                            }
                        },
                        "aws": {
                            "function_name": "xray123",
                            "region": "us-east-1",
```

```
                        "operation": "Invoke",
                        "resource_names": [
                            "xray123"
                        ]
                    },
                    "namespace": "aws"
                }
            ]
        },
        "Id": "a1b2c3d4a1b2c3d4"
    }
```

Arbeiten mit HTTP-APIs

REST-APIs und HTTP-APIs sind beide RESTful-API-Produkte. REST-APIs unterstützen mehr Funktionen als HTTP-APIs. HTTP-APIs sind mit minimalen Funktionen ausgestattet, sodass sie zu einem niedrigeren Preis angeboten werden können. Weitere Informationen finden Sie unter the section called “Auswahl zwischen REST-APIs und HTTP-APIs ”.

Sie können HTTP-APIs verwenden, um Anfragen an AWS Lambda Funktionen oder an jeden routbaren HTTP-Endpunkt zu senden. Beispielsweise können Sie eine HTTP-API erstellen, die in eine Lambda-Funktion auf dem Backend integriert wird. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion und gibt die Antwort der Funktion an den Client zurück.

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

Sie können HTTP-APIs mithilfe der AWS Management Console AWS CLI, der APIs oder SDKs erstellen. AWS CloudFormation

Themen

- Entwickeln einer HTTP-API in API Gateway
- HTTP-APIs zum Aufruf durch Kunden veröffentlichen
- HTTP-API schützen
- HTTP-API überwachen
- Fehlerbehebung bei Problemen mit HTTP-APIs

Entwickeln einer HTTP-API in API Gateway

Dieser Abschnitt enthält Details zu den API Gateway-Funktionen, die Sie bei der Entwicklung Ihrer API Gateway-APIs benötigen.

Während Sie Ihre API Gateway-API entwickeln, entscheiden Sie sich für eine Reihe von Merkmalen Ihrer API. Diese Eigenschaften hängen davon ab, wofür Ihre API verwendet werden soll. So könnte es beispielsweise sein, dass Sie es nur bestimmten Clients gestatten möchten, die API aufzurufen. Vielleicht soll die API aber auch für alle verfügbar sein. Vielleicht benötigen Sie einen API-Aufruf, um eine Lambda-Funktion auszuführen, eine Datenbankabfrage durchzuführen oder eine Anwendung aufzurufen.

Themen

Erstellen einer HTTP-API

Um eine funktionale API erstellen zu können, benötigen Sie mindestens eine Route, eine Integration, eine Phase und eine Bereitstellung.

Die folgenden Beispiele zeigen, wie Sie eine API mit einer AWS Lambda oder HTTP-Integration, einer Route und einer Standardstufe erstellen, die so konfiguriert ist, dass Änderungen automatisch bereitgestellt werden.

In diesem Leitfaden wird davon ausgegangen, dass Sie bereits mit API Gateway und Lambda vertraut sind. Einen ausführlicheren Leitfaden finden Sie unter Erste Schritte.

Themen

Erstellen Sie eine HTTP-API mithilfe der AWS Management Console

1. Öffnen Sie die API Gateway-Konsole.
2. Wählen Sie Create API (API erstellen) aus.
3. Wählen Sie unter HTTP-API die Option Entwickeln aus.
4. Wählen Sie Add integration (Integration hinzufügen) und wählen Sie dann eine AWS Lambda -Funktion oder geben Sie einen HTTP-Endpunkt ein.
5. Geben Sie im Feld Name einen Namen für Ihre API ein.
6. Wählen Sie Review and create.

7. Wählen Sie Create aus.

Jetzt kann Ihre API aufgerufen werden. Sie können Ihre API testen, indem Sie die Aufruf-URL in einem Browser eingeben oder cURL verwenden.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Erstellen Sie eine HTTP-API mithilfe der AWS CLI

Sie können Quick Create verwenden, um eine API mit einer Lambda- oder HTTP-Integration, eine Catch-All-Standardroute und eine Standardphase zu erstellen, die für die automatische Bereitstellung von Änderungen konfiguriert ist. Der folgende Befehl verwendet Quick Create, um eine API zu erstellen, die in eine Lambda-Funktion im Backend integriert werden kann.

> **Note**
>
> Um eine Lambda-Integration aufzurufen, muss API Gateway über die erforderlichen Berechtigungen verfügen. Sie können eine ressourcenbasierte Richtlinie oder eine IAM-Rolle verwenden, um API-Gateway Berechtigungen zum Aufrufen einer Lambda-Funktion zu erteilen. Weitere Informationen finden Sie unter AWS Lambda Berechtigungen im AWS Lambda Entwicklerhandbuch.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
 arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Jetzt kann Ihre API aufgerufen werden. Sie können Ihre API testen, indem Sie die Aufruf-URL in einem Browser eingeben oder cURL verwenden.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Arbeiten mit Routen für HTTP-APIs

Routen leiten eingehende API-Anfragen an Backend-Ressourcen. Routen bestehen aus zwei Teilen: einer HTTP-Methode und einem Ressourcenpfad, zum Beispie, `GET /pets`. Sie können spezifische HTTP-Methoden für Ihre Route definieren. Sie können auch die Methode `ANY` verwenden,

um alle Methoden abzugleichen, die Sie nicht für eine Ressource definiert haben. Sie können eine `$default`-Route erstellen, die als Catch-All für Anfragen fungiert, die nicht mit anderen Routen übereinstimmen.

> **Note**
>
> API Gateway dekodiert URL-kodierte Anforderungsparameter vor deren Übergabe an Ihre Backend-Integration.

Arbeiten mit Pfadvariablen

Sie können Pfadvariablen in HTTP-API-Routen verwenden.

Beispielsweise fängt die Route `GET /pets/{petID}` eine `GET`-Anforderung ab, die ein Client an `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6` sendet.

Eine gierige Pfadvariable fängt alle untergeordneten Ressourcen einer Route ab. Um eine gierige Pfadvariable zu erstellen, fügen Sie + dem Variablennamen hinzu, z. B. `{proxy+}`. Die gierige Pfadvariable muss am Ende des Ressourcenpfads stehen.

Arbeiten mit Abfragezeichenfolgenparametern

API Gateway sendet standardmäßig Abfragezeichenfolgenparameter an Ihre Backend-Integration, sofern sie in einer Anforderung an eine HTTP-API enthalten sind.

Wenn beispielsweise ein Client eine Anforderung an `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog` sendet, werden die Abfragezeichenfolgenparameter `?id=4&type=dog` an Ihre Integration gesendet.

Arbeiten mit der **`$default`**-Route

Die `$default`-Route fängt Anforderungen ab, die nicht explizit mit anderen Routen in Ihrer API übereinstimmen.

Wenn die `$default`-Route eine Anforderung empfängt, sendet API Gateway den vollständigen Anforderungspfad an die Integration. Beispielsweise können Sie eine API nur mit einer `$default`-Route erstellen und sie in der Methode `ANY` mit dem HTTP-Endpunkt `https://petstore-demo-endpoint.execute-api.com` integrieren. Wenn Sie eine Anforderung an `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout` senden, sendet API

Gateway eine Anforderung an `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Weitere Informationen zu HTTP-Integrationen finden Sie unter Arbeiten mit HTTP-Proxy-Integrationen für HTTP-APIs.

Weiterleiten von API-Anforderungen

Wenn ein Client eine API-Anforderung sendet, bestimmt API Gateway zuerst, an welche Stufe die Anforderung weitergeleitet werden soll. Wenn die Anforderung explizit mit einer Stufe übereinstimmt, sendet API Gateway die Anforderung an diese Stufe. Wenn keine Stufe vollständig mit der Anforderung übereinstimmt, sendet API Gateway die Anforderung an die `$default`-Stufe. Wenn es keine `$default` Phase gibt, kehrt die API zurück `{"message":"Not Found"}` und generiert keine CloudWatch Protokolle.

Nachdem Sie eine Stufe ausgewählt haben, wählt API Gateway eine Route aus. API Gateway wählt die Route mit der spezifischsten Übereinstimmung aus, wobei die folgenden Prioritäten verwendet werden:

1. Vollständige Übereinstimmung für eine Route und Methode.
2. Übereinstimmung für eine Route und eine Methode mit einer gierigen Pfadvariable (`{proxy+}`).
3. Die `$default`-Route.

Wenn keine Routen mit einer Anforderung übereinstimmen, gibt API Gateway `{"message":"Not Found"}` an den Client zurück.

Betrachten Sie beispielsweise eine API mit einer `$default`-Stufe und den folgenden Beispielrouten:

1. `GET /pets/dog/1`
2. `GET /pets/dog/{id}`
3. `GET /pets/{proxy+}`
4. `ANY /{proxy+}`
5. `$default`

In der folgenden Tabelle wird zusammengefasst, wie API Gateway Anforderungen an die Beispielrouten weiterleitet.

Anfrage	Ausgewählte Route	Erklärung
`GET https://api-id.execute-api. region.amazonaws.com/pets/dog/1`	`GET /pets/dog/1`	Die Anforderung entspricht vollständig dieser statischen Route.
`GET https://api-id.execute-api. region.amazonaws.com/pets/dog/2`	`GET /pets/dog/{id}`	Die Anforderung stimmt vollständig mit dieser Route überein.
`GET https://api-id.execute-api. region.amazonaws.com/pets/cat/1`	`GET /pets/{proxy+}`	Die Anforderung stimmt nicht vollständig mit einer Route überein. Die Route mit einer `GET`-Methode und einer gierigen Pfadvariablen fängt diese Anforderung ab.
`POST https://api-id.execute-api. region.amazonaws.com/test/5`	`ANY /{proxy+}`	Die Methode `ANY` entspricht allen Methoden, die Sie nicht für eine Route definiert haben. Routen mit gierigen Pfadvariablen haben eine höhere Priorität als die `$default`-Route.

Steuern und Verwalten des Zugriffs auf eine HTTP-API in API Gateway

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre HTTP-API:

- Lambda-Genehmiger verwenden Lambda-Funktionen, um den Zugriff auf APIs zu steuern. Weitere Informationen finden Sie unter Mit AWS Lambda Autorisierern für HTTP-APIs arbeiten.
- JWT-Genehmiger verwenden JSON-Web-Token, um den Zugriff auf APIs zu steuern. Weitere Informationen finden Sie unter Steuern des Zugriffs auf HTTP-APIs mit JWT-Genehmigern.

- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen. Sie können IAM-Rollen und -Richtlinien verwenden, um zu steuern, wer Ihre APIs erstellen und verwalten kann und wer sie aufrufen kann. Weitere Informationen finden Sie unter IAM-Genehmiger verwenden.

Mit AWS Lambda Autorisierern für HTTP-APIs arbeiten

Sie verwenden einen Lambda-Genehmiger, um den Zugriff auf Ihre HTTP-API mit einer Lambda-Funktion zu steuern. Wenn dann ein Client Ihre API aufruft, ruft API Gateway Ihre Lambda-Funktion auf. API Gateway verwendet die Antwort Ihrer Lambda-Funktion, um festzustellen, ob der Client auf Ihre API zugreifen kann.

Nutzlastformatversion

Die Nutzlastformatversion des Genehmigers gibt das Format der Daten an, die API Gateway an einen Lambda-Genehmiger sendet, und wie API Gateway die Antwort von Lambda interpretiert. Wenn Sie keine Version im Payload-Format angeben, wird standardmäßig die neueste Version AWS Management Console verwendet. Wenn Sie einen Lambda-Autorisierer mithilfe des AWS CLI, AWS CloudFormation, oder eines SDK erstellen, müssen Sie einen angeben. `authorizerPayloadFormatVersion` Die unterstützten Werte sind `1.0` und `2.0`.

Wenn Sie Kompatibilität mit REST-APIs benötigen, verwenden Sie Version `1.0`.

Die folgenden Beispiele zeigen die Struktur jeder Nutzlastformatversion.

2.0

```
{
  "version": "2.0",
  "type": "REQUEST",
  "routeArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": ["user1", "123"],
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": ["cookie1", "cookie2"],
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
```

```
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "IP",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "pathParameters": { "parameter1": "value1" },
  "stageVariables": { "stageVariable1": "value1", "stageVariable2": "value2" }
}
```

1.0

```
{
  "version": "1.0",
  "type": "REQUEST",
```

```
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": "user1,123",
  "authorizationToken": "user1,123",
  "resource": "/request",
  "path": "/request",
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
```

```
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

Antwortformat des Lambda-Genehmigers

Die Nutzlastformatversion bestimmt auch die Struktur der Antwort, die von Ihrer Lambda-Funktion zurückgeben werden muss.

Lambda-Funktionsantwort für Format 1.0

Wenn Sie Formatversion `1.0` auswählen, müssen Lambda-Genehmiger eine IAM-Richtlinie zurückgeben, die den Zugriff auf Ihre API-Route zulässt oder verweigert. Sie können die IAM-Standardrichtliniensyntax in der Richtlinie verwenden. Beispiele für IAM-Richtlinien finden Sie unter the section called “ Kontrollieren des Zugriffs für den API-Aufruf”. Mit können Sie Kontexteigenschaften an Lambda-Integrationen oder -Zugriffsprotokolle übergebe `$context.authorizer.`*`property`*. Das `context`-Objekt ist optional und `claims` ist ein reservierter Platzhalter, der nicht als Kontextobjekt verwendet werden kann. Weitere Informationen hierzu finden Sie unter the section called “Protokollierungsvariablen”.

Example

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
```

```
}
```

Lambda-Funktionsantwort für Format 2.0

Wenn Sie Formatversion `2.0` auswählen, können Sie einen booleschen Wert oder eine IAM-Richtlinie zurückgeben, die die IAM-Standardrichtliniensyntax Ihrer Lambda-Funktion verwendet. Um einen booleschen Wert zurückzugeben, aktivieren Sie einfache Antworten für den Genehmiger. Die folgenden Beispiele veranschaulichen das Format, das Sie für die Rückgabe Ihrer Lambda-Funktion codieren müssen. Das Objekt `context` ist optional. Mit können Sie Kontexteigenschaften an Lambda-Integrationen oder -Zugriffsprotokolle übergebe `$context.authorizer.`*`property`*. Weitere Informationen hierzu finden Sie unter the section called “Protokollierungsvariablen”.

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Beispiel für Lambda-Genehmiger-Funktionen

Die Lambda-Funktionen des folgenden Node.js-Beispiels veranschaulichen die erforderlichen Antwortformate, die Sie von Ihrer Lambda-Funktion für Formatversion `2.0` der Nutzlast zurückgeben müssen.

Simple response - Node.js

```
export const handler = async(event) => {
    let response = {
        "isAuthorized": false,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": true,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    };

    if (event.headers.authorization === "secretToken") {
        console.log("allowed");
        response = {
            "isAuthorized": true,
            "context": {
                "stringKey": "value",
                "numberKey": 1,
                "booleanKey": true,
                "arrayKey": ["value1", "value2"],
                "mapKey": {"value1": "value2"}
            }
        };
    }

    return response;

};
```

Simple response - Python

```
import json
```

```
def lambda_handler(event, context):
    response = {
        "isAuthorized": False,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                "isAuthorized": True,
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

IAM policy - Node.js

```
export const handler = async(event) => {
  if (event.headers.authorization == "secretToken") {
    console.log("allowed");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
```

```
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Allow",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
  else {
    console.log("denied");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Deny",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
};
```

IAM policy - Python

```
import json
```

```
def lambda_handler(event, context):
    response = {
        # The principal user identification associated with the token sent by
        # the client.
        "principalId": "abcdef",
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event["routeArn"]
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                # The principal user identification associated with the token
                # sent by the client.
                "principalId": "abcdef",
                "policyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Action": "execute-api:Invoke",
                        "Effect": "Allow",
                        "Resource": event["routeArn"]
                    }]
                },
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
```

```
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

Identitätsquellen

Sie können optional Identitätsquellen für einen Lambda-Genehmiger angeben. Identitätsquellen geben den Speicherort der Daten an, die für die Autorisierung einer Anforderung erforderlich sind. Beispielsweise können Sie Header- oder Abfragezeichenfolgenwerte als Identitätsquellen angeben. Wenn Sie Identitätsquellen angeben, müssen Clients diese in die Anforderung aufnehmen. Wenn die Anforderung des Clients die Identitätsquellen nicht enthält, ruft API Gateway Ihren Lambda-Genehmiger nicht auf, und der Client erhält einen `401`-Fehler. Die folgenden Identitätsquellen werden unterstützt:

Auswahlausdrücke

Typ	Beispiel	Hinweise
Header-Wert	$request.header.*Name*	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.
Abfragezeichenfolgenwert	$request.querystring.*Name*	Abfragezeichenfolgennamen unterscheiden zwischen Groß- und Kleinschreibung.
Kontextvariable	$context.*Variablenname*	Der Wert einer unterstützten Kontextvariablen.
Stufenvariable	$stageVariables.*Variablenname*	Der Wert einer Stufenvariablen.

Zwischenspeichern der Antworten des Genehmigers

Sie können das Caching für einen Lambda-Autorisierer aktivieren, indem Sie einen angeben. authorizerResultTtlInSeconds Wenn für einen Genehmiger Zwischenspeicherung aktiviert ist, verwendet API Gateway die Identitätsquellen des Genehmigers als Cache-Schlüssel. Wenn ein Client dieselben Parameter in Identitätsquellen innerhalb der konfigurierten TTL angibt, verwendet API Gateway das zwischengespeicherte Genehmiger-Ergebnis, anstatt Ihre Lambda-Funktion aufzurufen.

Um das Caching zu aktivieren, muss der Genehmiger über mindestens eine Identitätsquelle verfügen.

Wenn Sie einfache Antworten für einen Genehmiger aktivieren, erlaubt oder verweigert die Antwort des Genehmigers alle API-Anforderungen, die mit den zwischengespeicherten Identitätsquellwerten übereinstimmen, vollständig. Um detailliertere Berechtigungen zu erhalten, deaktivieren Sie einfache Antworten und geben Sie eine IAM-Richtlinie zurück.

Standardmäßig verwendet API Gateway die zwischengespeicherte Genehmiger-Antwort für alle Routen einer API, die den Genehmiger verwenden. Um Antworten pro Route zwischenzuspeichern, fügen Sie den Identitätsquellen Ihres Genehmigers `$context.routeKey` hinzu.

Erstellen eines Lambda-Genehmigers

Wenn Sie einen Lambda-Genehmiger erstellen, geben Sie die Lambda-Funktion an, die API Gateway verwenden soll. Sie müssen API Gateway die Berechtigung zum Aufrufen der Lambda-Funktion erteilen, indem Sie entweder die Ressourcenrichtlinie der Funktion oder eine IAM-Rolle verwenden. In diesem Beispiel aktualisieren wir die Ressourcenrichtlinie für die Funktion, so dass sie API Gateway die Berechtigung erteilt, unsere Lambda-Funktion aufzurufen.

```
aws apigatewayv2 create-authorizer \
    --api-id abcdef123 \
    --authorizer-type REQUEST \
    --identity-source '$request.header.Authorization' \
    --name lambda-authorizer \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \
    --authorizer-payload-format-version '2.0' \
    --enable-simple-responses
```

Der folgende Befehl gewährt API Gateway die Berechtigung zum Aufrufen Ihrer Lambda-Funktion. Wenn API Gateway nicht zum Aufrufen Ihrer Funktion berechtigt ist, erhalten Clients einen `500 Internal Server Error`.

```
aws lambda add-permission \
    --function-name my-authorizer-function \
    --statement-id apigateway-invoke-permissions-abc123 \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-
id/authorizers/authorizer-id"
```

Nachdem Sie einen Genehmiger erstellt und API Gateway die Berechtigung zum Aufrufen erteilt haben, aktualisieren Sie Ihre Route, um den Genehmiger zu verwenden.

```
aws apigatewayv2 update-route \
    --api-id abcdef123 \
    --route-id acd123 \
    --authorization-type CUSTOM \
    --authorizer-id def123
```

Fehlerbehebung für Lambda-Genehmiger

Wenn API Gateway Ihren Lambda-Genehmiger nicht aufrufen kann oder Ihr Lambda-Genehmiger eine Antwort in einem ungültigen Format zurückgibt, erhalten Clients einen `500 Internal Server Error`.

Um Fehler zu beheben, aktivieren Sie die Zugriffsprotokollierung für Ihre API-Stufe. Fügen Sie die `$context.authorizer.error`-Protokollierungsvariable in Ihr Protokollformat ein.

Wenn die Protokolle darauf hinweisen, dass API Gateway keine Berechtigung zum Aufrufen Ihrer Funktion besitzt, aktualisieren Sie die Ressourcenrichtlinie Ihrer Funktion oder stellen Sie eine IAM-Rolle bereit, um die API Gateway-Berechtigung zum Aufrufen des Genehmigers zu erteilen.

Wenn die Protokolle darauf hinweisen, dass Ihre Lambda-Funktion eine ungültige Antwort zurückgibt, überprüfen Sie, ob Ihre Lambda-Funktion eine Antwort im erforderlichen Format zurückgibt.

Steuern des Zugriffs auf HTTP-APIs mit JWT-Genehmigern

Sie können JSON Web Tokens (JWTs) als Teil der Frameworks OpenID Connect (OIDC) und OAuth 2.0 verwenden, um den Clientzugriff auf Ihre APIs einzuschränken.

Wenn Sie einen JWT-Genehmiger für eine Route Ihrer API konfigurieren, validiert API Gateway die JWTs, die Clients mit API-Anfragen übermitteln. API Gateway genehmigt oder lehnt Anfragen

basierend auf der Token-Validierung und optional Bereichen im Token ab. Wenn Sie Bereiche für eine Route konfigurieren, muss das Token mindestens einen der Bereiche der Route enthalten.

Sie können unterschiedliche Genehmiger für jede Route einer API konfigurieren oder denselben Genehmiger für mehrere Routen verwenden.

> **Note**
>
> Es gibt keinen Standardmechanismus, um JWT-Zugriffstoken von anderen Arten von JWTs wie beispielsweise OpenID Connect ID-Token zu unterscheiden. Wir empfehlen, Ihre Routen so zu konfigurieren, dass Autorisierungsbereiche erforderlich sind, es sei denn, Sie benötigen ID-Token für die API-Autorisierung Sie können Ihre JWT-Genehmiger auch so konfigurieren, dass Aussteller oder Zielgruppen erforderlich sind, die Ihr Identitätsanbieter nur bei der Ausgabe von JWT-Zugriffstoken verwendet.

Autorisieren von API-Anfragen mit einem JWT-Autorisierer

API Gateway verwendet den folgenden allgemeinen Workflow, um Anfragen an Routen zu autorisieren, die für die Verwendung eines JWT-Genehmigers konfiguriert sind.

1. Überprüfen Sie die `identitySource` auf ein Token. Der `identitySource` kann nur das Token oder das Token mit dem Präfix `Bearer` enthalten.
2. Decodieren Sie das Token.
3. Überprüfen Sie den Algorithmus und die Signatur des Tokens mit dem vom des Ausstellers abgerufenen öffentlichen Schlüsse `jwks_uri`. Derzeit werden nur RSA-basierte Algorithmen unterstützt. API Gateway kann den öffentlichen Schlüssel zwei Stunden lang zwischenspeichern. Es hat sich bewährt, beim Wechseln von Schlüsseln eine bestimmte Frist einzuräumen, während der sowohl der alte als auch der neue Schlüssel gültig sind.
4. Validieren Sie die Ansprüche. API Gateway evaluiert die folgenden Token-Ansprüche:
   - `kid` – Das Token muss einen Header-Anspruch haben, der mit dem Schlüssel in dem `jwks_uri` übereinstimmt, der das Token signiert hat.
   - `iss` – Muss mit dem für den Genehmiger konfigurierten `issuer` übereinstimmen.
   - `aud` oder `client_id` – Muss mit einem der für den Genehmiger konfigurierten `audience`-Einträge übereinstimmen. API Gateway validiert `client_id` nur, wenn `aud` es nicht vorhanden ist. Wenn beide `aud` und vorhanden `client_id` sind, bewertet `aud` API Gateway.
   - `exp` – Muss nach der aktuellen Uhrzeit in UTC liegen.

- `nbf` – muss vor der aktuellen Uhrzeit in UTC liegen.
- `iat` – Muss vor der aktuellen Uhrzeit in UTC liegen.
- `scope` oder `scp` – Das Token muss mindestens einen der Bereiche in den `authorizationScopes` der Route enthalten.

Wenn einer dieser Schritte fehlschlägt, lehnt API Gateway die API-Anfrage ab.

Nach der Validierung des JWT übergibt API Gateway die Ansprüche im Token an die Integration der API-Route. Backend-Ressourcen, wie beispielsweise Lambda-Funktionen, können auf die JWT-Ansprüche zugreifen. Wenn das JWT beispielsweise den Identitätsanspruch `emailID` enthält, ist er für eine Lambda-Integration in `$event.requestContext.authorizer.jwt.claims.emailID` verfügbar. Weitere Informationen zur Nutzlast, die API Gateway an Lambda-Integrationen sendet, finden Sie unter the section called “AWS Lambda Integrationen”.

Erstellen eines JWT-Genehmigers

Bevor Sie einen JWT-Genehmiger erstellen, müssen Sie eine Clientanwendung bei einem Identitätsanbieter registrieren. Sie müssen darüber hinaus eine HTTP-API erstellt haben. Beispiele zum Erstellen einer HTTP-API finden Sie unter Erstellen einer HTTP-API.

Erstellen Sie mithilfe der Konsole einen JWT-Autorisierer

Die folgenden Schritte zeigen, wie Sie einen JWT-Autorisierer mithilfe der Konsole erstellen.

Um einen JWT-Autorisierer mit der Konsole zu erstellen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie im Hauptnavigationsbereich die Option Autorisierung aus.
4. Wählen Sie die Registerkarte Autorisatoren verwalten.
5. Wählen Sie Erstellen.
6. Wählen Sie als Autorisierungstyp die Option JWT aus.
7. Konfigurieren Sie Ihren JWT-Autorisierer und geben Sie eine Identitätsquelle an, die die Quelle des Tokens definiert.
8. Wählen Sie Erstellen.

Erstellen Sie einen JWT-Autorisierer mit dem AWS CLI

Der folgende AWS CLI Befehl erstellt einen JWT-Autorisierer. Bei `jwt-configuration` geben Sie das `Audience` und `Issuer` als Ihren Identitätsanbieter an. Wenn Sie Amazon Cognito als Identitätsanbieter verwenden, `IssuerUrl` ist `https://cognito-idp.`*`us-east-2`*`.amazonaws.com/`*`userPoolID`* dies der Fall.

```
aws apigatewayv2 create-authorizer \
    --name authorizer-name \
    --api-id api-id \
    --authorizer-type JWT \
    --identity-source '$request.header.Authorization' \
    --jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Erstellen Sie einen JWT-Autorisierer mit AWS CloudFormation

Die folgende AWS CloudFormation Vorlage erstellt eine HTTP-API mit einem JWT-Autorisierer, der Amazon Cognito als Identitätsanbieter verwendet.

Die Ausgabe der AWS CloudFormation Vorlage ist eine URL für eine von Amazon Cognito gehostete Benutzeroberfläche, über die sich Kunden registrieren und anmelden können, um ein JWT zu erhalten. Nachdem sich ein Client angemeldet hat, wird der Client mit einem Zugriffstoken in der URL zu Ihrer HTTP-API umgeleitet. Um die API mit dem Zugriffstoken aufzurufen, ändern Sie das # in der URL in a, ? um das Token als Abfragezeichenfolgenparameter zu verwenden.

Beispiel für AWS CloudFormation eine Vorlage

```
AWSTemplateFormatVersion: '2010-09-09'
Description: |
  Example HTTP API with a JWT authorizer. This template includes an Amazon Cognito user
 pool as the issuer for the JWT authorizer
  and an Amazon Cognito app client as the audience for the authorizer. The outputs
 include a URL for an Amazon Cognito hosted UI where clients can
  sign up and sign in to receive a JWT. After a client signs in, the client is
 redirected to your HTTP API with an access token
  in the URL. To invoke the API with the access token, change the '#' in the URL to a
 '?' to use the token as a query string parameter.

Resources:
  MyAPI:
    Type: AWS::ApiGatewayV2::Api
    Properties:
```

```
      Description: Example HTTP API
      Name: api-with-auth
      ProtocolType: HTTP
      Target: !GetAtt MyLambdaFunction.Arn
  DefaultRouteOverrides:
    Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
    Properties:
      ApiId: !Ref MyAPI
      Route:
        AuthorizationType: JWT
        AuthorizerId: !Ref JWTAuthorizer
  JWTAuthorizer:
    Type: AWS::ApiGatewayV2::Authorizer
    Properties:
      ApiId: !Ref MyAPI
      AuthorizerType: JWT
      IdentitySource:
        - '$request.querystring.access_token'
      JwtConfiguration:
        Audience:
        - !Ref AppClient
        Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}
      Name: test-jwt-authorizer
  MyLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Runtime: nodejs18.x
      Role: !GetAtt FunctionExecutionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
              const response = {
                  statusCode: 200,
                  body: JSON.stringify('Hello from the ' + event.routeKey + ' route!'),
              };
              return response;
          };
  APIInvokeLambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref MyLambdaFunction
      Action: lambda:InvokeFunction
      Principal: apigateway.amazonaws.com
```

```
      SourceArn: !Sub arn:${AWS::Partition}:execute-api:${AWS::Region}:
${AWS::AccountId}:${MyAPI}/$default/$default
  FunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
              - lambda.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: http-api-user-pool
      AutoVerifiedAttributes:
        - email
      Schema:
        - Name: name
          AttributeDataType: String
          Mutable: true
          Required: true
        - Name: email
          AttributeDataType: String
          Mutable: false
          Required: true
  AppClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      AllowedOAuthFlows:
        - implicit
      AllowedOAuthScopes:
        - aws.cognito.signin.user.admin
        - email
        - openid
        - profile
      AllowedOAuthFlowsUserPoolClient: true
      ClientName: api-app-client
      CallbackURLs:
```

```
        - !Sub https://${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
      ExplicitAuthFlows:
        - ALLOW_USER_PASSWORD_AUTH
        - ALLOW_REFRESH_TOKEN_AUTH
      UserPoolId: !Ref UserPool
      SupportedIdentityProviders:
        - COGNITO
  HostedUI:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
      Domain: !Join
        - '-'
        - - !Ref MyAPI
          - !Ref AppClient
      UserPoolId: !Ref UserPool
Outputs:
  SignupURL:
    Value: !Sub https://${HostedUI}.auth.${AWS::Region}.amazoncognito.com/login?
client_id=${AppClient}&response_type=token&scope=email+profile&redirect_uri=https://
${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
```

Aktualisieren Sie eine Route, um einen JWT-Autorisierer zu verwenden

Sie können die Konsole, das oder ein AWS SDK verwenden AWS CLI, um eine Route so zu aktualisieren, dass sie einen JWT-Autorisierer verwendet.

Aktualisieren Sie eine Route mithilfe der Konsole, um einen JWT-Autorisierer zu verwenden

Die folgenden Schritte zeigen, wie Sie eine Route für die Verwendung des JWT-Autorisierers mithilfe der Konsole aktualisieren.

Um einen JWT-Autorisierer mit der Konsole zu erstellen

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie im Hauptnavigationsbereich die Option Autorisierung aus.
4. Wählen Sie eine Methode aus, wählen Sie dann Ihren Autorisierer aus dem Drop-down-Menü aus und wählen Sie Autorisierer anhängen aus.

Aktualisieren Sie eine Route, um einen JWT-Autorisierer zu verwenden, indem Sie AWS CLI

Der folgende Befehl aktualisiert eine Route für die Verwendung eines JWT-Autorisierers mithilfe von. AWS CLI

```
aws apigatewayv2 update-route \
   --api-id api-id  \
   --route-id route-id  \
   --authorization-type JWT \
   --authorizer-id authorizer-id \
   --authorization-scopes user.email
```

IAM-Genehmiger verwenden

Sie können die IAM-Autorisierung für HTTP-API-Routen aktivieren. Wenn die IAM-Autorisierung aktiviert ist, müssen Clients Signature Version 4 (Sigv4) verwenden, um ihre Anfragen mit AWS Anmeldeinformationen zu signieren. API Gateway ruft Ihre API-Route nur auf, wenn der Client über die `execute-api`-Berechtigung für die Route verfügt.

Die IAM-Autorisierung für HTTP-APIs ähnelt der für REST-APIs.

> **Note**
>
> Ressourcenrichtlinien werden derzeit nicht für HTTP-APIs unterstützt.

Beispiele für IAM-Richtlinien, die Clients die Berechtigung zum Aufrufen von APIs gewähren, finden Sie unter the section called “ Kontrollieren des Zugriffs für den API-Aufruf”.

Aktivieren der IAM-Autorisierung für eine Route

Der folgende AWS CLI Befehl aktiviert die IAM-Autorisierung für eine HTTP-API-Route.

```
aws apigatewayv2 update-route \
    --api-id abc123 \
    --route-id abcdef \
    --authorization-type AWS_IAM
```

Konfigurieren von Integrationen für HTTP-APIs

Integrationen verbinden eine Route mit Backend-Ressourcen. HTTP-APIs unterstützen Lambda-Proxy-, AWS Service- und HTTP-Proxyintegrationen. Beispielsweise können Sie eine `POST`-Anfrage an die `/signup`-Route Ihrer API so konfigurieren, dass sie in eine Lambda-Funktion integriert wird, die die Registrierung von Kunden übernimmt.

Themen

- Arbeiten mit AWS Lambda Proxy-Integrationen für HTTP-APIs
- Arbeiten mit HTTP-Proxy-Integrationen für HTTP-APIs
- Arbeiten mit AWS Serviceintegrationen für HTTP-APIs
- Arbeiten mit privaten Integrationen für HTTP-APIs

Arbeiten mit AWS Lambda Proxy-Integrationen für HTTP-APIs

Eine Lambda-Proxy-Integration ermöglicht es Ihnen, eine API-Route in eine Lambda-Funktion zu integrieren. Wenn ein Client Ihre API aufruft, sendet API Gateway die Anfrage an die Lambda-Funktion und gibt die Antwort der Funktion an den Client zurück. Beispiele zum Erstellen einer HTTP-API finden Sie unter Erstellen einer HTTP-API.

Nutzlastformatversion

Die Payload-Formatversion spezifiziert das Format des Ereignisses, das API Gateway an eine Lambda-Integration sendet, und wie API Gateway die Antwort von Lambda interpretiert. Wenn Sie keine Version im Payload-Format angeben, wird standardmäßig die neueste Version AWS Management Console verwendet. Wenn Sie eine Lambda-Integration mithilfe des AWS CLI, AWS CloudFormation, oder eines SDK erstellen, müssen Sie a `payloadFormatVersion` angeben. Die unterstützten Werte sind `1.0` und `2.0`.

Weitere Informationen zum Einstellen von finden Sie unter `payloadFormatVersion` create-integration. Weitere Informationen darüber, wie Sie den `payloadFormatVersion` Wert einer vorhandenen Integration ermitteln können, finden Sie unter get-integration

Unterschiede im Payload-Format

Die folgende Liste zeigt die Unterschiede zwischen den Versionen im Payload-Format `1.0` und im `2.0` Payload-Format:

- Das Format `2.0` hat keine `multiValueHeaders`- oder `multiValueQueryStringParameters`-Felder. Doppelte Header werden mit Kommas kombiniert und in das `headers`-Feld aufgenommen. Doppelte Abfragezeichenfolgen werden mit Kommas kombiniert und in das `queryStringParameters`-Feld aufgenommen.
- Das Format `2.0` hat`rawPath`. Wenn Sie eine API-Zuordnung verwenden, um Ihre Stufe mit einem benutzerdefinierten Domainnamen zu verbinden, `rawPath` wird der API-Zuordnungswert nicht bereitgestellt. Verwenden Sie das Format `1.0` und`path`, um auf die API-Zuordnung für Ihren benutzerdefinierten Domainnamen zuzugreifen.
- Das Format `2.0` enthält ein neues `cookies`-Feld. Alle Cookie-Header in der Anforderung werden mit Kommas kombiniert und dem `cookies`-Feld hinzugefügt. In der Antwort an den Client wird jedes Cookie zu einem `set-cookie`-Header.

Struktur des Payload-Formats

Die folgenden Beispiele zeigen die Struktur jeder Nutzlastformatversion. Alle Headernamen sind klein geschrieben.

2.0

```
{
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": [
    "cookie1",
    "cookie2"
  ],
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "api-id",
    "authentication": {
```

```
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "authorizer": {
      "jwt": {
        "claims": {
          "claim1": "value1",
          "claim2": "value2"
        },
        "scopes": [
          "scope1",
          "scope2"
        ]
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "192.0.2.1",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "body": "Hello from Lambda",
  "pathParameters": {
    "parameter1": "value1"
  },
  "isBase64Encoded": false,
  "stageVariables": {
```

```
    "stageVariable1": "value1",
    "stageVariable2": "value2"
  }
}
```

1.0

```
{
  "version": "1.0",
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
```

```
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "192.0.2.1",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
```

```
}
```

Antwortformat der Lambda-Funktion

Die Nutzlastformatversion bestimmt die Struktur der Antwort, die Ihre Lambda-Funktion zurückgeben muss.

Lambda-Funktionsantwort für Format 1.0

Bei der `1.0` Formatversion müssen Lambda-Integrationen eine Antwort im folgenden JSON-Format zurückgeben:

Example

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
    "body": "..."
}
```

Lambda-Funktionsantwort für Format 2.0

Bei der `2.0`-Formatversion kann API Gateway das Antwortformat für Sie ableiten. API Gateway geht von den folgenden Annahmen aus, wenn Ihre Lambda-Funktion gültigen JSON-Code und keinen zurückgib `statusCode`:

- `isBase64Encoded` ist `false`.
- `statusCode` ist `200`.
- `content-type` ist `application/json`.
- `body` ist die Antwort der Funktion.

Die folgenden Beispiele zeigen die Ausgabe einer Lambda-Funktion und die Interpretation von API Gateway.

Lambda-Funktionsausgabe	API Gateway-Interpretation
`"Hello from Lambda!"`	`{ "isBase64Encoded": false, "statusCode": 200, "body": "Hello from Lambda!", "headers": { "content-type": "application/json" } }`
`{ "message": "Hello from Lambda!" }`	`{ "isBase64Encoded": false, "statusCode": 200, "body": "{ \"message\": \"Hello from Lambda!\" }", "headers": { "content-type": "application/json" } }`

Um die Antwort anzupassen, sollte Ihre Lambda-Funktion eine Antwort im folgenden Format zurückgeben.

```
{
    "cookies" : ["cookie1", "cookie2"],
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "body": "Hello from Lambda!"
}
```

Arbeiten mit HTTP-Proxy-Integrationen für HTTP-APIs

Eine HTTP-Proxy-Integration ermöglicht es Ihnen, eine API-Route mit einem öffentlich routingfähigen HTTP-Endpunkt zu verbinden. Bei diesem Integrationstyp wird die gesamte Anforderung und Antwort von API Gateway zwischen dem Frontend und dem Backend übergeben.

Geben Sie die URL eines öffentlich routingfähigen HTTP-Endpunkts an, um eine HTTP-Proxy-Integration zu erstellen.

HTTP-Proxy-Integration mit Pfadvariablen

Sie können Pfadvariablen in HTTP-API-Routen verwenden.

Beispielsweise fängt die Route `/pets/{petID}` Anforderungen an `/pets/6` ab. Sie können Pfadvariablen im Integrations-URI referenzieren, um den Inhalt der Variablen an eine Integration zu senden. Ein Beispiel ist `/pets/extendedpath/{petID}`.

Sie können gierige Pfadvariablen verwenden, um alle untergeordneten Ressourcen einer Route zu erfassen. Um eine gierige Pfadvariable zu erstellen, fügen Sie + dem Variablennamen hinzu, z. B. `{proxy+}`.

Um eine Route mit einer HTTP-Proxy-Integration einzurichten, die alle Anforderungen abfängt, erstellen Sie eine API-Route mit einer gierigen Pfadvariable (z. B, `/parent/{proxy+}`). Integrieren Sie die Route mit einem HTTP-Endpunkt (z. B. `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) für die ANY-Methode. Die gierige Pfadvariable muss am Ende des Ressourcenpfads stehen.

Arbeiten mit AWS Serviceintegrationen für HTTP-APIs

Sie können Ihre HTTP-API mithilfe erstklassiger Integrationen in AWS Dienste integrieren. Eine erstklassige Integration verbindet eine HTTP-API-Route mit einer AWS -Service-API. Wenn ein Client eine Route aufruft, die durch eine erstklassige Integration unterstützt wird, ruft API Gateway eine AWS Service-API für Sie auf. Sie können beispielsweise erstklassige Integrationen verwenden, um eine Nachricht an eine Amazon Simple Queue Service-Warteschlange zu senden oder eine AWS Step Functions Zustandsmaschine zu starten. Unterstützte Serviceaktionen finden Sie unter the section called “AWS Referenz zu Serviceintegrationen”.

Mapping von Anforderungsparametern

Erstklassige Integrationen verfügen über erforderliche und optionale Parameter. Sie müssen alle erforderlichen Parameter zum Erstellen einer Integration konfigurieren. Sie können statische Werte oder Mapping-Parameter verwenden, die zur Laufzeit dynamisch ausgewertet werden. Eine vollständige Liste der unterstützten Integrationen und Parameter finden Sie unter the section called “AWS Referenz zu Serviceintegrationen”.

Parameter-Mapping

Typ	Beispiel	Hinweise
Header-Wert	$request.header.*Name*	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. `"header1": "value1,value2"` .
Abfragezeichenfolgenwert	$request.querystring.*Name*	Abfragezeichenfolgennamen unterscheiden zwischen Groß- und Kleinschreibung. API Gateway kombiniert mehrere Werte mit Kommas, z. B. `"querystring1": "Value1,Value2"` .
Path-Parameter	$request.path.*Name*	Der Wert eines Pfadparameters in der Anforderung. Beispiel: Wenn die Route `/pets/{petId}` ist, können Sie den Parameter `petId` aus der Anforderung mit *`$request.path.petid`* zuordnen.
Übergeben des Anforderungstexts	$request.body	API Gateway übergibt den gesamten Anforderungstext.
Anforderungstext	$request.body.*Name*	Ein JSON-Pfadausdruck. Rekursiver Abstieg (`$request.body..` *`name`*) und Filterausdrücke (`?(`*`expression`*` )`) werden nicht unterstützt.

Typ	Beispiel	Hinweise
		Note Wenn Sie einen JSON-Pfad angeben, schneidet API Gateway den Anfragetext auf 100 KB ab und wendet dann den Auswahlausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an `$request.body` .
Kontextvariable	$context.*`Variablenname`*	Der Wert einer unterstützten Kontextvariablen.
Stufenvariable	$stageVariables.*`Variablenname`*	Der Wert einer Stufenvariablen.
Statischer Wert	*`string`*	Ein konstanter Wert.

Erstellen Sie einer erstklassigen Integration

Bevor Sie eine erstklassige Integration erstellen, müssen Sie eine IAM-Rolle erstellen, die API Gateway Gateway-Berechtigungen zum Aufrufen der AWS Serviceaktion gewährt, mit der Sie die Integration durchführen. Weitere Informationen finden Sie unter Erstellen einer Rolle für einen AWS -Service.

Um eine erstklassige Integration zu erstellen, wählen Sie eine unterstützte AWS Serviceaktion aus, z. B. `SQS-SendMessage` konfigurieren Sie die Anforderungsparameter, und stellen Sie eine Rolle bereit, die API Gateway Gateway-Berechtigungen zum Aufrufen der integrierten AWS Service-API gewährt. Je nach Integration-Subtyp sind unterschiedliche Anforderungsparameter

erforderlich. Weitere Informationen hierzu finden Sie unter the section called “AWS Referenz zu Serviceintegrationen”.

Der folgende AWS CLI Befehl erstellt eine Integration, die eine Amazon SQS SQS-Nachricht sendet.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-subtype SQS-SendMessage \
    --integration-type AWS_PROXY \
    --payload-format-version 1.0 \
    --credentials-arn arn:aws:iam::123456789012:role/apigateway-sqs \
    --request-parameters '{"QueueUrl": "$request.header.queueUrl", "MessageBody":
 "$request.body.message"}'
```

Erstellen Sie eine erstklassige Integration mit AWS CloudFormation

Das folgende Beispiel zeigt ein AWS CloudFormation Snippet, das eine `/{source}/{detailType}` Route mit einer erstklassigen Integration mit Amazon erstellt. EventBridge

Der Parameter `Source` wird dem Pfadparameter `{source}`, der `DetailType` dem Pfadparameter `{DetailType}` und der Parameter `Detail` dem Anforderungstext zugeordnet.

Das Snippet zeigt weder den Event Bus noch die IAM-Rolle, die API-Gateway-Berechtigungen zum Aufrufen der Aktion `PutEvents` gewährt.

```
Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      AuthorizationType: None
      RouteKey: 'POST /{source}/{detailType}'
      Target: !Join
        - /
        - - integrations
          - !Ref Integration
  Integration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref HttpApi
      IntegrationType: AWS_PROXY
      IntegrationSubtype: EventBridge-PutEvents
      CredentialsArn: !GetAtt EventBridgeRole.Arn
      RequestParameters:
```

```
        Source: $request.path.source
        DetailType: $request.path.detailType
        Detail: $request.body
        EventBusName: !GetAtt EventBus.Arn
      PayloadFormatVersion: "1.0"
```

Integration-Subtypreferenz

Die folgenden Integration-Subtypen werden für HTTP-APIs unterstützt.

Integration-Subtypen

- EventBridge-PutEvents
- SQS- SendMessage
- SQS- ReceiveMessage
- SQS- DeleteMessage
- SQS- PurgeQueue
- AppConfig-GetConfiguration
- Kinesis- PutRecord
- StepFunctions-StartExecution
- StepFunctions-StartSyncExecution
- StepFunctions-StopExecution

EventBridge-PutEvents

Sendet benutzerdefinierte Ereignisse an Amazon, EventBridge damit sie mit Regeln abgeglichen werden können.

EventBridge- PutEvents 1.0

Parameter	Erforderlich
Detail	True
DetailType	True
Source	Wahr
Zeit	False

Parameter	Erforderlich
EventBusName	False
Ressourcen	Falsch
Region	False
TraceHeader	False

Weitere Informationen finden Sie PutEventsin der Amazon EventBridge API-Referenz.

SQS- SendMessage

Übergibt eine Nachricht an die angegebene Warteschlange

SQS- 1.0 SendMessage

Parameter	Erforderlich
QueueUrl	True
MessageBody	True
DelaySeconds	False
MessageAttributes	False
MessageDeduplicationId	False
MessageGroupId	False
MessageSystemAttributes	False
Region	False

Weitere Informationen finden Sie SendMessagein der Amazon Simple Queue Service API-Referenz.

SQS- ReceiveMessage

Ruft eine oder mehrere Nachrichten (bis zu 10) aus der angegebenen Warteschlange ab.

SQS- 1.0 ReceiveMessage

Parameter	Erforderlich
QueueUrl	True
AttributeNames	False
MaxNumberOfMessages	False
MessageAttributeNames	False
ReceiveRequestAttemptId	False
VisibilityTimeout	False
WaitTimeSeconds	False
Region	False

Weitere Informationen finden Sie ReceiveMessagein der Amazon Simple Queue Service API-Referenz.

SQS- DeleteMessage

Löscht die angegebene Nachricht aus der angegebenen Warteschlange

SQS- 1.0 DeleteMessage

Parameter	Erforderlich
ReceiptHandle	True
QueueUrl	True
Region	False

Weitere Informationen finden Sie DeleteMessagein der Amazon Simple Queue Service API-Referenz.

SQS- PurgeQueue

Löscht alle Nachrichten in der angegebenen Warteschlange.

SQS- 1.0 PurgeQueue

Parameter	Erforderlich
QueueUrl	True
Region	False

Weitere Informationen finden Sie PurgeQueuein der Amazon Simple Queue Service API-Referenz.

AppConfig-GetConfiguration

Fordert Informationen zu einer Konfiguration an.

AppConfig- GetConfiguration 1.0

Parameter	Erforderlich
Anwendung	Wahr
Umgebung	Wahr
Konfiguration	True
ClientId	True
ClientConfigurationVersion	False
Region	False

Weitere Informationen finden Sie GetConfigurationin der AWS AppConfig API-Referenz.

Kinesis- PutRecord

Schreibt einen einzelnen Datensatz in einen Amazon Kinesis-Daten-Stream.

Kinesis- 1.0 PutRecord

Parameter	Erforderlich
StreamName	True

Parameter	Erforderlich
Daten	True
PartitionKey	True
SequenceNumberForOrdering	False
ExplicitHashKey	False
Region	False

Weitere Informationen finden Sie PutRecordin der Amazon Kinesis Data Streams API-Referenz.

StepFunctions-StartExecution

Startet eine Ausführung des Zustandsautomaten.

StepFunctions- 1.0 StartExecution

Parameter	Erforderlich
StateMachineArn	True
Name	Falsch
Eingabe	Falsch
Region	False

Weitere Informationen finden Sie StartExecutionin der AWS Step Functions API-Referenz.

StepFunctions-StartSyncExecution

Startet eine synchrone Ausführung des Zustandsautomaten.

StepFunctions- StartSyncExecution 1.0

Parameter	Erforderlich
StateMachineArn	True

Parameter	Erforderlich
Name	Falsch
Eingabe	Falsch
Region	False
TraceHeader	False

Weitere Informationen finden Sie StartSyncExecutionin der AWS Step Functions API-Referenz.

StepFunctions-StopExecution

Stoppt eine Ausführung.

StepFunctions- StopExecution 1.0

Parameter	Erforderlich
ExecutionArn	True
Ursache	Falsch
Fehler	Falsch
Region	False

Weitere Informationen finden Sie StopExecutionin der AWS Step Functions API-Referenz.

Arbeiten mit privaten Integrationen für HTTP-APIs

Private Integrationen ermöglichen es Ihnen, API-Integrationen mit privaten Ressourcen in einer VPC zu erstellen, z. B. Application Load Balancer oder containerbasierte Amazon ECS-Anwendungen.

Sie können Ihre Ressourcen in einer VPC für den Zugriff von Clients außerhalb der VPC bereitstellen, indem Sie private Integrationen verwenden. Sie können den Zugriff auf Ihre API steuern, indem Sie eine der von API Gateway unterstützten Autorisierungsmethoden verwenden.

Um eine private Integration zu erstellen, müssen Sie zunächst einen VPC-Link erstellen. Weitere Informationen zu VPC-Links finden Sie unter Arbeiten mit VPC-Links für HTTP-APIs.

Nachdem Sie einen VPC-Link erstellt haben, können Sie private Integrationen einrichten, die eine Verbindung zu einem Application Load Balancer, Network Load Balancer oder Ressourcen herstellen, die bei einem Service registriert sind. AWS Cloud Map

Um eine private Integration zu erstellen, müssen alle Ressourcen demselben AWS Konto gehören (einschließlich Load Balancer oder AWS Cloud Map Service, VPC-Link und HTTP-API).

Standardmäßig verwendet der private Integrationsdatenverkehr das HTTP-Protokoll. Sie können eine `tlsConfig` angeben, wenn Sie den privaten Integrationsdatenverkehr für die Verwendung von HTTPS benötigen.

> **Note**
>
> Bei privaten Integrationen schließt API Gateway den Stufen-Teil des API-Endpunkts in die Anforderung an Ihre Backend-Ressourcen ein. Beispielsweise enthält eine Anforderung an die Stufe `test` einer API in der Anforderung an Ihre private Integration `test/`*`route-path`*. Um den Künstlernamen aus der Anforderung an Ihre Back-End-Ressourcen zu entfernen, verwenden Sie Parameterzuordnung, um den Anforderungspfad nach `$request.path` zu überschreiben.

Erstellen einer privaten Integration mit einem Application Load Balancer oder Network Load Balancer

Bevor Sie eine private Integration erstellen, müssen Sie einen VPC-Link erstellen. Weitere Informationen zu VPC-Links finden Sie unter Arbeiten mit VPC-Links für HTTP-APIs.

Um eine private Integration mit einem Application Load Balancer oder Network Load Balancer zu erstellen, erstellen Sie eine HTTP-Proxy-Integration, geben Sie den zu verwendenden VPC-Link an und stellen Sie den Listener-ARN des Load Balancers bereit.

Verwenden Sie den folgenden Befehl, um eine private Integration zu erstellen, die über einen VPC-Link eine Verbindung mit einem Load Balancer herstellt.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65
    --payload-format-version 1.0
```

Erstellen Sie mithilfe AWS Cloud Map von Service Discovery eine private Integration

Bevor Sie eine private Integration erstellen, müssen Sie einen VPC-Link erstellen. Weitere Informationen zu VPC-Links finden Sie unter Arbeiten mit VPC-Links für HTTP-APIs.

Bei Integrationen mit AWS Cloud Map verwendet `DiscoverInstances` API Gateway Ressourcen. Sie können Abfrageparameter verwenden, um bestimmte Ressourcen zu adressieren. Die Attribute der registrierten Ressourcen müssen IP-Adressen und Ports enthalten. API Gateway verteilt Anforderungen auf fehlerfreie Ressourcen, die von zurückgegeben werde `DiscoverInstances`. Weitere Informationen finden Sie DiscoverInstancesin der AWS Cloud Map API-Referenz.

> **Note**
>
> Wenn Sie Amazon ECS zum Ausfüllen von Einträgen verwenden AWS Cloud Map, müssen Sie Ihre Amazon ECS-Aufgabe so konfigurieren, dass sie SRV-Datensätze mit Amazon ECS Service Discovery verwendet, oder Amazon ECS Service Connect aktivieren. Weitere Informationen finden Sie unter Services miteinander verbinden im Amazon Elastic Container Service-Entwicklerhandbuch.

Um eine private Integration mit zu erstellen AWS Cloud Map, erstellen Sie eine HTTP-Proxyintegration, geben Sie den zu verwendenden VPC-Link an und geben Sie den ARN des AWS Cloud Map Dienstes an.

Verwenden Sie den folgenden Befehl, um eine private Integration zu erstellen, die AWS Cloud Map Service Discovery verwendet, um Ressourcen zu identifizieren.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY  \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:servicediscovery:us-east-2:123456789012:service/srv-id?
stage=prod&deployment=green_deployment
    --payload-format-version 1.0
```

Arbeiten mit VPC-Links für HTTP-APIs

Mit VPC-Links können Sie private Integrationen erstellen, die Ihre HTTP-API-Routen mit privaten Ressourcen in einer VPC verbinden, z. B. Application Load Balancer oder containerbasierte Amazon ECS-Anwendungen. Weitere Informationen zum Erstellen von privaten Integrationen finden Sie unter Arbeiten mit privaten Integrationen für HTTP-APIs.

Eine private Integration verwendet einen VPC-Link, um Verbindungen zwischen API Gateway und zielgerichteten VPC-Ressourcen einzukapseln. Sie können VPC-Links über verschiedene Routen und APIs hinweg wiederverwenden.

Wenn Sie einen VPC-Link erstellen, erstellt und verwaltet API Gateway elastische Netzwerkschnittstellen für den VPC-Link in Ihrem Konto. Dieser Vorgang kann einige Minuten dauern. Wenn ein VPC-Link einsatzbereit ist, wechselt sein Status von `PENDING` zu `AVAILABLE`.

> **Note**
>
> Wenn 60 Tage lang kein Datenverkehr über den VPC-Link gesendet wird, wird er `INACTIVE`. Wenn sich ein VPC-Link in einem `INACTIVE`-Zustand befindet, werden alle Netzwerkschnittstellen des VPC-Links von API Gateway gelöscht. Dies führt dazu, dass API-Anforderungen, die von dem VPC-Link abhängen, fehlschlagen. Wenn API-Anforderungen wieder aufgenommen werden, stellt API Gateway Netzwerkschnittstellen wieder her. Es kann einige Minuten dauern, bis die Netzwerkschnittstellen erstellt und der VPC-Link reaktiviert wird. Sie können den VPC-Linkstatus verwenden, um den Status Ihres VPC-Links zu überwachen.

Erstellen Sie einen VPC-Link mit dem AWS CLI

Verwenden Sie den folgenden Befehl, um einen VPC-Link zu erstellen. Um einen VPC-Link zu erstellen, müssen alle beteiligten Ressourcen demselben AWS Konto gehören.

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \
    --subnet-ids subnet-aaaa subnet-bbbb \
    --security-group-ids sg1234 sg5678
```

> **Note**
>
> VPC-Links sind unveränderlich. Nachdem Sie einen VPC-Link erstellt haben, können Sie seine Subnetze oder Sicherheitsgruppen nicht ändern.

Löschen Sie einen VPC-Link mit dem AWS CLI

Verwenden Sie den folgenden Befehl, um einen VPC-Link zu löschen.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Verfügbarkeit nach Region

VPC-Links für HTTP-APIs werden in den folgenden Regionen und Availability Zones unterstützt:

Name der Region	Region	Unterstützte Availability Zones
USA Ost (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
USA Ost (Nord-Virginia)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
USA West (Nordkalifornien)	us-west-1	usw1-az1, usw1-az3
USA West (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asien-Pazifik (Hongkong)	ap-east-1	ape1-az2, ape1-az3
Asien-Pazifik (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asien-Pazifik (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asien-Pazifik (Singapur)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3
Asien-Pazifik (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3

Name der Region	Region	Unterstützte Availability Zones
Asien-Pazifik (Tokio)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Kanada (Zentral)	ca-central-1	cac1-az1, cac1-az2
Europa (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europa (Irland)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europa (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europa (Paris)	eu-west-3	euw3-az1, euw3-az3
Europa (Stockholm)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Naher Osten (Bahrain)	me-south-1	mes1-az1, mes1-az2, mes1-az3
Südamerika (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Konfigurieren von CORS für eine HTTP-API

Cross-Origin Resource Sharing (CORS) ist eine Browser-Sicherheitsfunktion, die HTTP-Anfragen einschränkt, die von im Browser ausgeführten Skripts eingeleitet werden. Wenn Sie nicht auf Ihre API zugreifen können und eine Fehlermeldung erhalten, die `Cross-Origin Request Blocked`

enthält, müssen Sie möglicherweise CORS aktivieren. Weitere Informationen finden Sie unter Was ist CORS? .

CORS ist in der Regel zum Erstellen von Webanwendungen erforderlich, die auf APIs zugreifen, die auf einer anderen Domäne oder einem anderen Ursprungsserver gehostet werden. Sie können CORS aktivieren, um Anfragen an Ihre API von einer Webanwendung zuzulassen, die auf einer anderen Domäne gehostet wird. Wenn Ihre API beispielsweise auf `https://{api_id}.execute-api.{region}.amazonaws.com/` gehostet wird und Sie Ihre API über eine Webanwendung aufrufen möchten, die auf `example.com` gehostet wird, muss Ihre API CORS unterstützen.

Wenn Sie CORS für eine API konfigurieren, sendet API Gateway automatisch eine Antwort auf Preflight-OPTIONS-Anfragen, selbst wenn keine OPTIONS-Route für Ihre API konfiguriert ist. Bei einer CORS-Anfrage fügt API Gateway die konfigurierten CORS-Header zur Antwort einer Integration hinzu.

> **Note**
>
> Wenn Sie CORS für eine API konfigurieren, ignoriert API Gateway CORS-Header, die von Ihrer Backend-Integration zurückgegeben werden.

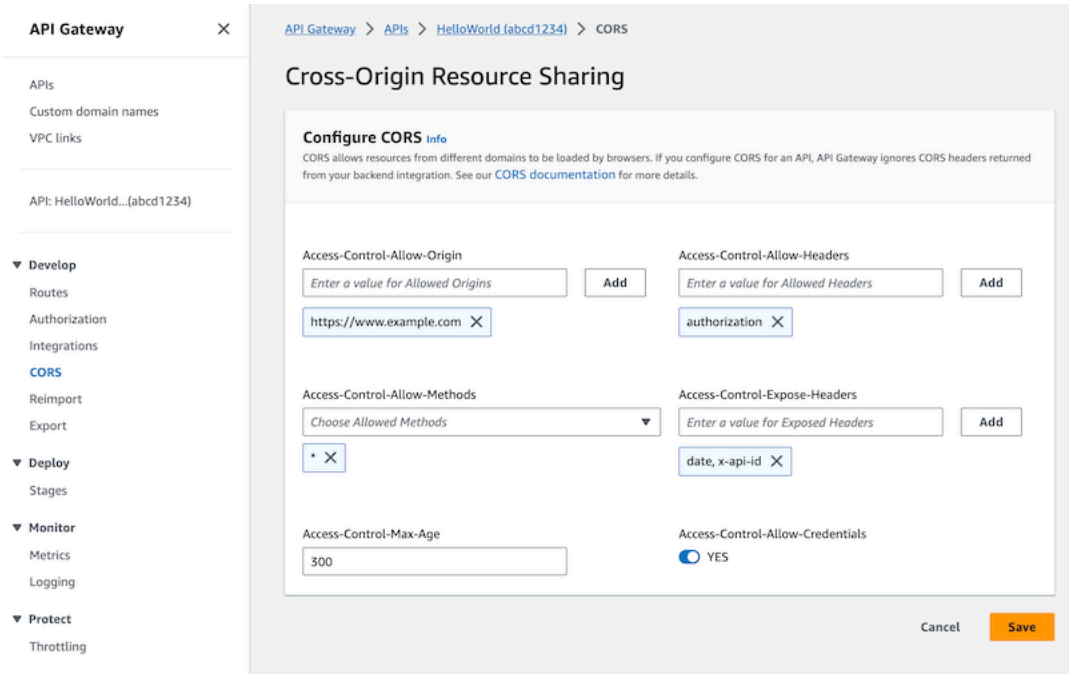
Sie können die folgenden Parameter in einer CORS-Konfiguration angeben. Um diese Parameter mithilfe der HTTP-API-Konsole des API Gateway hinzuzufügen, wählen Sie Add (Hinzufügen), nachdem Sie Ihren Wert eingegeben haben.

CORS-Header	CORS-Konfigurationseigenschaft	Beispielwerte
Access-Control-Allow-Origin	allowOrigins	• `https://www.example.com` • `*` (alle Ursprünge zulassen) • `https://*` (einen beliebigen Ursprung zulassen, der mit https:// beginnt)

CORS-Header	CORS-Konfigurationseigenschaft	Beispielwerte
		• `http://*` (einen beliebigen Ursprung zulassen, der mit http:// beginnt)
Access-Control-Allow-Credentials	allowCredentials	true
Access-Control-Expose-Headers	exposeHeaders	Datum x-api-id, *
Access-Control-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	GET, POST, DELETE, *
Access-Control-Allow-Headers	allowHeaders	Autorisierung, *

Um CORS-Header zurückzugeben, muss Ihre Anfrage einen `origin`-Header enthalten.

Ihre CORS-Konfiguration sieht möglicherweise wie folgt aus:



Konfiguration von CORS für eine HTTP-API mit einer **`$default`** Route und einem Authorizer

Sie können CORS aktivieren und die Autorisierung für jede Route einer HTTP-API konfigurieren. Wenn Sie CORS und die Autorisierung für die `$default`-Route aktivieren, gibt es einige besondere Überlegungen. Die `$default`-Route fängt Anforderungen für alle Methoden und Routen ab, die Sie nicht explizit definiert haben, einschließlich `OPTIONS`-Anforderungen. Fügen Sie Ihrer API zur Unterstützung von nicht autorisierten `OPTIONS`-Anforderungen eine `OPTIONS /{proxy+}`-Route hinzu, die keine Autorisierung erfordert. Die Route `OPTIONS /{proxy+}` hat eine höhere Priorität als die Route `$default`. Dadurch können Clients ohne Autorisierung `OPTIONS`-Anforderungen an Ihre API senden. Weitere Informationen zu Weiterleitungsprioritäten finden Sie unter Weiterleiten von API-Anforderungen.

Konfigurieren von CORS für eine HTTP-API mithilfe der AWS CLI

Sie können den folgenden update-api-Befehl verwenden, um CORS-Anfragen von zu aktivieren. `https://www.example.com`

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://
www.example.com"
```

Weitere Informationen finden Sie unter CORS in der API-Referenz für Amazon API Gateway Version 2.

Transformieren von API-Anforderungen und -Antworten

Sie können API-Anforderungen von Clients ändern, bevor sie Ihre Backend-Integrationen erreichen. Sie können auch die Antwort von Integrationen ändern, bevor API Gateway die Antwort an Clients zurückgibt. Sie verwenden das Parameter-Mapping, um API-Anforderungen und -Antworten für HTTP-APIs zu ändern. Um das Parameter-Mapping zu verwenden, geben Sie zu ändernde API-Anforderungs- oder Antwortparameter an und wie diese Parameter geändert werden sollen.

Transformieren von API-Anforderungen

Sie verwenden Anforderungsparameter, um Anforderungen zu ändern, bevor sie Ihre Backend-Integrationen erreichen. Sie können Header, Abfragezeichenfolgen oder den Anforderungspfad ändern.

Anforderungsparameter sind eine Schlüssel-Wert-Zuordnung. Der Schlüssel gibt den Speicherort des zu ändernden Anforderungsparameters und an, wie er geändert werden soll. Der Wert gibt die neuen Daten für den Parameter an.

Die folgende Tabelle zeigt unterstützte Schlüssel.

Parameter-Mapping-Schlüssel

Typ	Syntax
Header	`append|overwrite|remove:header.` *`headername`*
Abfragezeichenfolge	`append|overwrite|remove:querystring.` *`querystring-name`*
Pfad	`overwrite:path`

Die folgende Tabelle zeigt unterstützte Werte, die Sie Parametern zuordnen können.

Anforderungsparameter-Mapping-Werte

Typ	Syntax	Hinweise
Header-Wert	$request.header.*name* oder ${request.header.*name*}	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterscheiden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. `"header1" : "value1,value2"` . Einige Header sind reserviert. Weitere Informationen hierzu finden Sie unter the section called “Reservierte Header”.
Abfragezeichenfolgenwert	$request.querystring.*name* oder ${request.querystring.*name*}	Abfragezeichenfolgennamen unterscheiden zwischen Groß- und Kleinschreibung.

Typ	Syntax	Hinweise
		API Gateway kombiniert mehrere Werte mit Kommas, z. B. `"querystring1" "Value1,Value2"` .
Anforderungstext	$request.body.*name* oder ${request.body.*name*}	Ein JSON-Pfadausdruck. Rekursiver Abstieg (`$request.body..name`) und Filterausdrücke (`?(expression)`) werden nicht unterstützt. **Note** Wenn Sie einen JSON-Pfad angeben, schneidet API Gateway den Anfragetext auf 100 KB ab und wendet dann den Auswahlausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an `$request.body` .
Anforderungspfad	$request.path oder ${request.path}	Der Anforderungspfad ohne den Namen der Stufe.

Typ	Syntax	Hinweise
Path-Parameter	$request.path.*name* oder ${request.path.*name*}	Der Wert eines Pfadparameters in der Anforderung. Beispiel: Wenn die Route `/pets/{petId}` ist, können Sie den Parameter `petId` aus der Anforderung mit `$request.path.petId` zuordnen.
Kontextvariable	$context.*variableName* oder ${context.*variableName* }	Der Wert einer Kontextvariablen. **Note** NEs werden ausschließlich die Sonderzeichen `.` und `_` unterstützt.
Stufenvariable	$stageVariables.*variableName* oder ${stageVariables.*variableName* }	Der Wert einer Stufenvariablen.
Statischer Wert	*string*	Ein konstanter Wert.

Note

Um mehrere Variablen in einem Auswahlausdruck zu verwenden, schließen Sie die Variable in Klammern ein. z. B. `${request.path.name} ${request.path.id}`.

Transformieren von API-Antworten

Sie verwenden Antwortparameter, um die HTTP-Antwort einer Backend-Integration zu transformieren, bevor Sie die Antwort an Clients zurückgeben. Sie können Header oder den Statuscode einer Antwort ändern, bevor API Gateway die Antwort an Clients zurückgibt.

Sie konfigurieren die Antwortparameter für jeden Statuscode, den Ihre Integration zurückgibt. Antwortparameter sind eine Schlüssel-Wert-Zuordnung. Der Schlüssel gibt den Speicherort des zu ändernden Anforderungsparameters und an, wie er geändert werden soll. Der Wert gibt die neuen Daten für den Parameter an.

Die folgende Tabelle zeigt unterstützte Schlüssel.

Antwortparameter-Mapping-Schlüssel

Typ	Syntax
Header	`append|overwrite|remove:header.` *`headername`*
Statuscode	`overwrite:statuscode`

Die folgende Tabelle zeigt unterstützte Werte, die Sie Parametern zuordnen können.

Antwortparameter-Mapping-Werte

Typ	Syntax	Hinweise
Header-Wert	$response.header.*Name* oder $ {response.header.*name*}	Bei Header-Namen wird nicht zwischen Groß- und Kleinschreibung unterscheiden. API Gateway kombiniert mehrere Headerwerte mit Kommas, z. B. `"header1" : "value1,value2"` . Einige Header sind reserviert. Weitere Informationen hierzu finden Sie unter the section called “Reservierte Header”.

Typ	Syntax	Hinweise
Antworttext	$response.body.*name* oder ${response.body.*name*}	Ein JSON-Pfadausdruck. Rekursiver Abstieg (`$response.body..name`) und Filterausdrücke (`?(expression)`) werden nicht unterstützt. **Note** Wenn Sie einen JSON-Pfad angeben, kürzt API Gateway den Antworttext auf 100 KB und wendet dann den Auswahlausdruck an. Um Payloads mit mehr als 100 KB zu senden, geben Sie an `$response.body`.
Kontextvariable	$context.*variableName* oder ${context.*variableName*}	Der Wert einer unterstützten Kontextvariablen.
Stufenvariable	$stageVariables.*variableName* oder ${stageVariables.*variableName*}	Der Wert einer Stufenvariablen.
Statischer Wert	*string*	Ein konstanter Wert.

> **Note**
>
> Um mehrere Variablen in einem Auswahlausdruck zu verwenden, schließen Sie die Variable in Klammern ein. z. B. `${request.path.name} ${request.path.id}`.

Reservierte Header

Die folgenden Header sind reserviert. Sie können keine Anforderungs- oder Antwort-Mappings für diese Header konfigurieren.

- access-control-*
- apigw-*
- Autorisierung
- Verbindung
- Content-Encoding
- Content-Length
- Content-Location
- Forwarded
- Keep-Alive
- Urspung
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-*
- x-amzn-*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

Beispiele

In den folgenden AWS CLI Beispielen werden Parameterzuordnungen konfiguriert. AWS CloudFormation Beispielvorlagen finden Sie unter. GitHub

Hinzufügen eines Headers zu einer API-Anforderung

Im folgenden Beispiel wird eine Kopfzeile mit dem Namen `header1` zu einer API-Anforderung hinzugefügt, bevor sie Ihre Backend-Integration erreicht. API Gateway füllt den Header mit der Anforderungs-ID auf.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header1": "$context.requestId" }'
```

Umbenennen eines Anforderungsheaders

Im folgenden Beispiel wird ein Anforderungsheader von `header1` in `header2` umbenannt.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header2": "$request.header.header1",
 "remove:header.header1": "''"}'
```

Ändern der Antwort aus einer Integration

Im folgenden Beispiel werden Antwortparameter für eine Integration konfiguriert. Wenn die Integrationen einen 500-Statuscode zurückgeben, ändert API Gateway den Statuscode auf 403 und fügt der Antwort `header11` hinzu. Wenn die Integration einen 404-Statuscode zurückgibt, fügt API Gateway der Antwort einen `error`-Header hinzu.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
```

```
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --response-parameters '{"500" : {"append:header.header1": "$context.requestId",
 "overwrite:statuscode" : "403"}, "404" : {"append:header.error" :
 "$stageVariables.environmentId"}  }'
```

Entfernen konfigurierter Parameter-Mappings

Der folgende Beispielbefehl entfernt zuvor konfigurierte Anforderungsparameter für `append:header.header1`. Außerdem werden zuvor konfigurierte Antwortparameter für einen 200-Statuscode entfernt.

```
aws apigatewayv2 update-integration \
    --api-id abcdef123 \
    --integration-id hijk456 \
    --request-parameters '{"append:header.header1" : ""}' \
    --response-parameters '{"200" : {}}'
```

Arbeiten mit OpenAPI-Definitionen für HTTP-APIs

Sie können Ihre HTTP-API definieren, indem Sie eine OpenAPI 3.0-Definitionsdatei verwenden. Anschließend können Sie die Definition in API Gateway importieren, um eine API zu erstellen. Weitere Informationen zu API Gateway-Erweiterungen für OpenAPI finden Sie unter Erweiterungen für OpenAPI.

Importieren einer HTTP-API

Sie können eine HTTP-API erstellen, indem Sie eine OpenAPI 3.0-Definitionsdatei importieren.

Um von einer REST-API zu einer HTTP-API zu migrieren, können Sie Ihre REST-API als OpenAPI 3.0-Definitionsdatei exportieren. Importieren Sie dann die API-Definition als HTTP-API. Weitere Informationen zum Exportieren einer REST-API finden Sie unter REST-API von API Gateway importieren.

> **Note**
>
> HTTP-APIs unterstützen dieselben AWS Variablen wie REST-APIs. Weitere Informationen hierzu finden Sie unter AWS Variablen für den OpenAPI-Import.

Importieren von Validierungsinformationen

Beim Importieren einer API stellt API Gateway drei Kategorien von Validierungsinformationen bereit.

Informationen

Eine Eigenschaft ist gemäß der OpenAPI-Spezifikation gültig, diese Eigenschaft wird für HTTP-APIs jedoch nicht unterstützt.

Beispielsweise erzeugt das folgende OpenAPI 3.0-Snippet Informationen zum Import, da HTTP-APIs die Anfragevalidierung nicht unterstützen. API Gateway ignoriert die Felder `requestBody` und `schema`.

```
"paths": {
  "/": {
    "get": {
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
        "payloadFormatVersion": "1.0"
      },
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Body"
            }
          }
        }
      }
    }
  }
  ...
},
"components": {
  "schemas": {
    "Body": {
      "type": "object",
      "properties": {
        "key": {
          "type": "string"
        }
```

```
      }
    }
    ...
  }
  ...
}
```

Warnung

Eine Eigenschaft oder Struktur ist gemäß der OpenAPI-Spezifikation ungültig, blockiert die API-Erstellung jedoch nicht. Sie können angeben, ob API Gateway diese Warnungen ignorieren und mit der API-Erstellung fortfahren oder die API-Erstellung bei Warnungen beenden soll.

Das folgende OpenAPI 3.0-Dokument erzeugt Warnungen beim Import, da HTTP-APIs nur Lambda-Proxy- und HTTP-Proxy-Integrationen unterstützen.

```
"x-amazon-apigateway-integration": {
  "type": "AWS",
  "httpMethod": "POST",
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
  "payloadFormatVersion": "1.0"
}
```

Fehler

Die OpenAPI-Spezifikation ist ungültig oder fehlerhaft. API Gateway kann keine Ressourcen aus dem fehlerhaften Dokument erstellen. Sie müssen die Fehler beheben und es anschließend erneut versuchen.

Die folgende API-Definition erzeugt Fehler beim Import, da HTTP-APIs nur die OpenAPI 3.0-Spezifikation unterstützen.

```
{
  "swagger": "2.0.0",
  "info": {
    "title": "My API",
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
    "version": "1.0"
  }
  ...
}
```

Ein weiteres Beispiel: Während OpenAPI es Benutzern ermöglicht, eine API mit mehreren Sicherheitsanforderungen zu definieren, die mit einer bestimmten Operation verbunden sind, wird dies von API Gateway nicht unterstützt. Jede Operation kann nur eine IAM-Autorisierung, einen Lambda-Genehmiger oder einen JWT-Genehmiger haben. Der Versuch, mehrere Sicherheitsanforderungen zu modellieren, führt zu einem Fehler.

Importieren Sie eine API mit dem AWS CLI

Der folgende Befehl importiert die OpenAPI 3.0-Definitionsdatei `api-definition.json` als HTTP-API.

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

Sie können das folgende Beispiel einer OpenAPI 3.0-Definition importieren, um eine HTTP-API zu erstellen.

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",
```

```
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pets"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      },
      "post": {
        "operationId": "Create Pet",
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/NewPet"
              }
            }
          },
          "required": true
        },
        "responses": {
```

```
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/NewPetResponse"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "HTTP_PROXY",
        "httpMethod": "POST",
        "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
        "payloadFormatVersion": 1.0
      }
    }
  },
  "/pets/{petId}": {
    "get": {
      "operationId": "Get Pet",
      "parameters": [
        {
          "name": "petId",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
```

```
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pet"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/
{petId}",
          "payloadFormatVersion": 1.0
        }
      }
    }
  },
  "x-amazon-apigateway-cors": {
    "allowOrigins": [
      "*"
    ],
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-amzm-header",
      "x-apigateway-header",
      "x-api-key",
      "authorization",
      "x-amz-date",
      "content-type"
    ]
  },
  "components": {
    "schemas": {
```

```
      "Pets": {
        "type": "array",
        "items": {
          "$ref": "#/components/schemas/Pet"
        }
      },
      "Empty": {
        "type": "object"
      },
      "NewPetResponse": {
        "type": "object",
        "properties": {
          "pet": {
            "$ref": "#/components/schemas/Pet"
          },
          "message": {
            "type": "string"
          }
        }
      },
      "Pet": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "type": {
            "type": "string"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "NewPet": {
        "type": "object",
        "properties": {
          "type": {
            "$ref": "#/components/schemas/PetType"
          },
          "price": {
            "type": "number"
          }
        }
```

```
      },
      "PetType": {
        "type": "string",
        "enum": [
          "dog",
          "cat",
          "fish",
          "bird",
          "gecko"
        ]
      }
    }
  }
}
```

Exportieren einer HTTP-API aus API Gateway

Nachdem Sie ein HTTP-API erstellt haben, können Sie eine OpenAPI 3.0-Definition Ihrer API aus API Gateway exportieren. Sie können entweder eine Stufe auswählen, die exportiert werden soll, oder die neueste Konfiguration Ihrer API exportieren. Sie können auch eine exportierte API-Definition nach API Gateway importieren, um eine weitere, identische API zu erstellen. Weitere Informationen zum Importieren von API-Definitionen finden Sie unter Importieren einer HTTP-API.

Exportieren Sie eine OpenAPI 3.0-Definition einer Phase mithilfe der CLI AWS

Der folgende Befehl exportiert eine OpenAPI-Definition einer API-Stufe namens `prod` in eine YAML-Datei namens `stage-definition.yaml`. Die exportierte Definitionsdatei enthält standardmäßig API Gateway-Erweiterungen.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type YAML  \
    --specification OAS30 \
    --stage-name prod \
    stage-definition.yaml
```

Exportieren Sie eine OpenAPI 3.0-Definition der neuesten Änderungen Ihrer API mithilfe der CLI AWS

Der folgende Befehl exportiert eine OpenAPI-Definition einer HTTP-API in eine JSON-Datei namens `latest-api-definition.json`. Da der Befehl keine Stufe angibt, exportiert API Gateway die

neueste Konfiguration Ihrer API, unabhängig davon, ob sie auf einer Stufe bereitgestellt wurde oder nicht. Die exportierte Definitionsdatei enthält keine API Gateway-Erweiterungen.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type JSON  \
    --specification OAS30 \
    --no-include-extensions \
    latest-api-definition.json
```

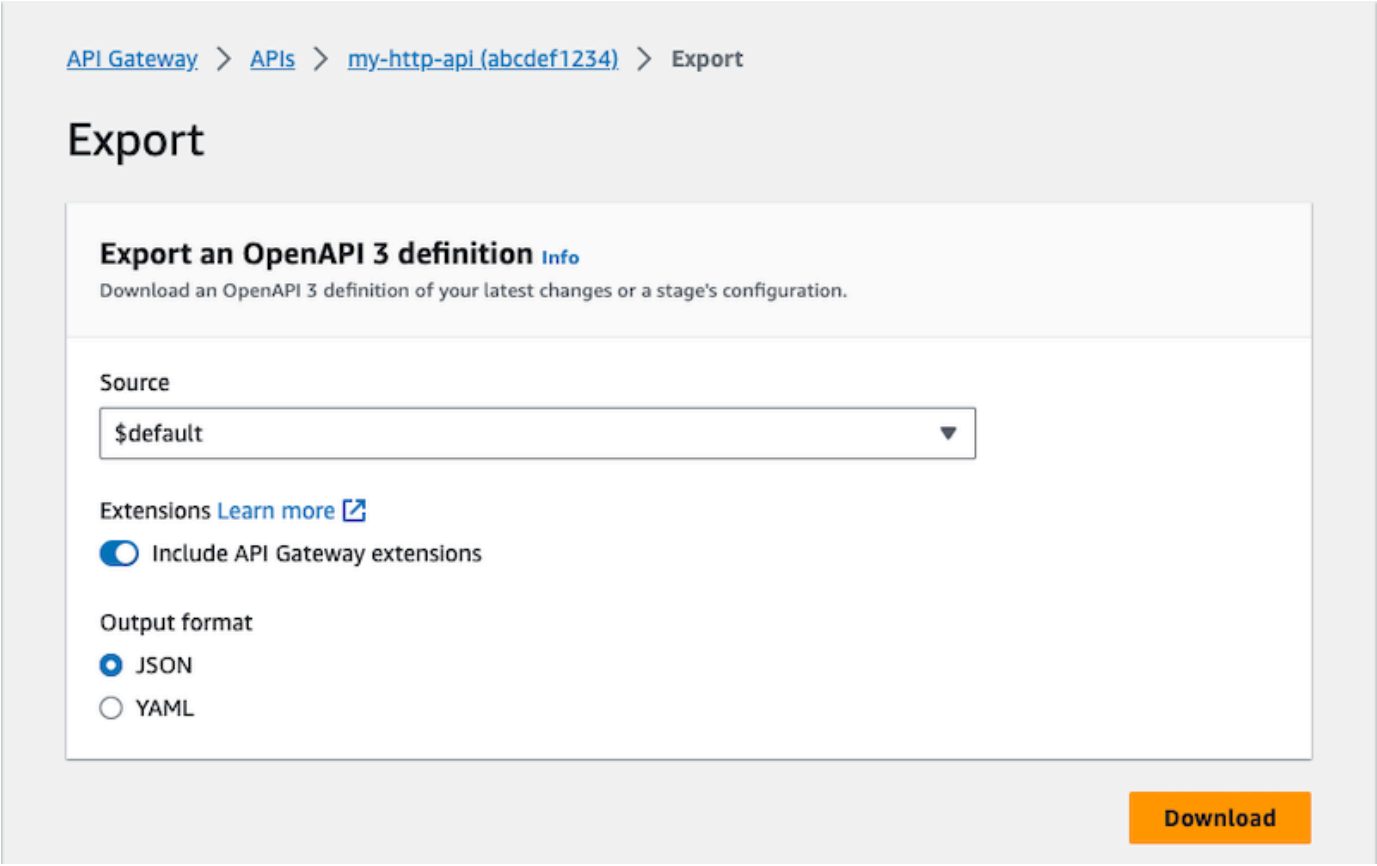
Weitere Informationen finden Sie unter ExportAPI in der API-Referenz für Amazon API Gateway Version 2.

Exportieren einer OpenAPI 3.0-Definition mithilfe der API-Gateway-Konsole

Das folgende Verfahren zeigt, wie Sie eine OpenAPI-Definition einer HTTP-API exportieren.

Exportieren einer OpenAPI 3.0-Definition mithilfe der API-Gateway-Konsole

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine HTTP-API.
3. Wählen Sie im Hauptnavigationsbereich unter Entwickeln die Option Exportieren aus.
4. Wählen Sie aus den folgenden Optionen, um Ihre API zu exportieren:



a. Wählen Sie unter Quelle eine Quelle für die OpenAPI 3.0-Definition aus. Sie können entweder eine Stufe auswählen, die exportiert werden soll, oder die neueste Konfiguration Ihrer API exportieren.

b. Aktivieren Sie API Gateway-Erweiterungen einschließen, um API-Gateway-Erweiterungen einzubeziehen.

c. Wählen Sie unter Ausgabeformat ein Ausgabeformat aus.

5. Wählen Sie Herunterladen aus.

HTTP-APIs zum Aufruf durch Kunden veröffentlichen

Sie können Stufen und benutzerdefinierte Domänennamen verwenden, um Ihre API zu veröffentlichen, die Clients aufrufen können.

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. `dev`, `prod`, `beta` oder `v2`). Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt. Sie können verschiedene Integrationen und Einstellungen für jede Stufe einer API konfigurieren.

Sie können benutzerdefinierte Domänennamen verwenden, um eine einfachere, intuitivere URL für Clients bereitzustellen, die Ihre API aufrufen können, als die Standard-UR, `https://api-id.execute-api.region.amazonaws.com/stage`.

> **Note**
>
> Um die Sicherheit Ihrer API-Gateway-APIs zu erhöhen, ist die `execute-api.{region}.amazonaws.com`-Domain in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-`-Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domain-Namen für Ihre API-Gateway-APIs einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Themen

Mit Stufen für HTTP-APIs arbeiten

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. `dev`, `prod`, `beta` oder `v2`). API-Stufen werden durch ihre API-ID und den Stufennamen identifiziert und sind in der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

Sie können eine `$default`-Phase erstellen, die beispielsweise von der Basis der URL Ihrer API bereitgestellt wird, z. B. `https://{api_id}.execute-api.{region}.amazonaws.com/`. Sie verwenden diese URL, um eine API-Phase aufzurufen.

Eine Bereitstellung ist ein Snapshot Ihrer API-Konfiguration. Nachdem Sie eine API in einer Phase bereitgestellt haben, kann sie von Clients aufgerufen werden. Sie müssen eine API bereitstellen, damit Änderungen wirksam werden. Wenn Sie automatische Bereitstellungen aktivieren, werden Änderungen an einer API automatisch für Sie freigegeben.

Stufenvariablen

Stufenvariablen sind Schlüssel-Werte-Paare, die Sie für eine Stufe einer HTTP-API definieren können. Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung verwendet werden.

Sie können beispielsweise eine Stufenvariable definieren und dann ihren Wert als HTTP-Endpunkt für eine HTTP-Proxy-Integration festlegen. Später können Sie den Endpunkt mithilfe des zugeordneten Stufenvariablennamens referenzieren. Auf diese Weise können Sie in jeder Stufe dieselbe API-Einrichtung mit einem anderen Endpunkt verwenden. In ähnlicher Weise können Sie Stufenvariablen verwenden, um für jede Phase Ihrer API eine andere AWS Lambda Funktionsintegration anzugeben.

> **Note**
>
> Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Genehmigers übergeben. Weitere Informationen hierzu finden Sie unter the section called “Antwortformat des Lambda-Genehmigers”.

Beispiele

Wenn Sie eine Stufenvariable zum Anpassen des HTTP-Integrationsendpunkts verwenden möchten, müssen Sie zunächst den Namen und den Wert der Stufenvariablen (z. B. `url`) mit dem Wert `example.com` festlegen. Richten Sie als Nächstes eine HTTP-Proxy-Integration ein. Anstatt die URL des Endpunkts einzugeben, können Sie API Gateway anweisen, den Wert der Stufenvariablen zu verwenden, **`http://${stageVariables.url}`**. Dieser Wert weist API Gateway an, Ihre Stufenvariable `${}` zur Laufzeit abhängig von der Stufe Ihrer API zu ersetzen.

Sie können auf ähnliche Weise auf Stufenvariablen verweisen, um einen Lambda-Funktionsnamen oder einen AWS Rollen-ARN anzugeben.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Dafür können Sie die AWS Command Line Interface (AWS CLI) verwenden.

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
```

```
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

API Gateway Stufenvariablenreferenz

HTTP-Integrations-URIs

Sie können eine Stufenvariable als Teil einer HTTP-Integrations-URI verwenden, wie in den folgenden Beispielen gezeigt.

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Lambda-Funktionen

Sie können eine Stufenvariable anstelle eines Integrationsnamens oder Alias für die Lambda-Funktion verwenden, wie in den folgenden Beispielen gezeigt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

AWS Anmeldeinformationen für die Integration

Sie können eine Stage-Variable als Teil eines ARN mit AWS Benutzer- oder Rollenanmeldedaten verwenden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Sicherheitsrichtlinie für HTTP-APIs

API Gateway setzt eine Sicherheitsrichtlinie von `TLS_1_2` für alle HTTP-API-Endpunkte durch.

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus TLS-Mindestversion und Verschlüsselungssammlungen, die von Amazon API Gateway angeboten werden. Das TLS-Protokoll behandelt Netzwerksicherheitsprobleme wie Manipulationen und Abhören zwischen einem Client und einem Server. Wenn Ihre Clients über die benutzerdefinierte Domäne einen TLS-Handshake mit Ihrer API ausführen, erzwingt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite-Optionen, die von Ihren Clients verwendet werden können. Diese Sicherheitsrichtlinie akzeptiert TLS 1.2- und TLS 1.3-Verkehr und lehnt TLS 1.0-Verkehr ab.

Unterstützte TLS-Protokolle und Chiffren für HTTP-APIs

In der folgenden Tabelle werden die unterstützten TLS-Protokolle und Chiffren für HTTP-APIs beschrieben.

Sicherheitsrichtlinie	TLS_1_2
TLS-Protokolle	
TLSv1.3	♦
TLSv1.2	♦
TLS-Chiffren	
TLS-AES-128-GCM-SHA256	♦
TLS-AES-256-GCM-SHA384	♦
TLS-CHACHA20-POLY1305-SHA256	♦

Sicherheitsrichtlinie	TLS_1_2
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für dieselben Chiffren. Eine Liste der Chiffriernamen finden Sie unter. the section called “OpenSSL- und RFC-Verschlüsselungsnamen”

Informationen zu REST-APIs und APIs WebSocket

Weitere Informationen zu REST-APIs und WebSocket APIs finden Sie unter the section called “Auswahl einer Sicherheitsrichtlinie” undthe section called “Sicherheitsrichtlinie für WebSocket APIs”.

Benutzerdefinierte Domänennamen für HTTP-APIs einrichten

Benutzerdefinierte Domänennamen sind einfachere und intuitivere URLs, die Sie Ihren API-Benutzern zur Verfügung stellen können.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

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

wo von API Gateway generiert `api-id` wird, `region` (AWS Region) wird von Ihnen bei der Erstellung der API angegeben und `stage` wird von Ihnen bei der Bereitstellung der API angegeben.

Der Hostname-Teil der URL (d. h. *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`) verweist auf einen API-Endpunkt. Der Standard-API-Endpunkt hat möglicherweise einen schwer zu merkenden, wenig benutzerfreundlichen Namen.

Mit benutzerdefinierten Domänennamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. `myservice`) auswählen, um die alternative URL Ihrer API zuzuordnen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

> **Note**
>
> Eine benutzerdefinierte Domäne kann mit REST-APIs und HTTP-APIs verknüpft werden. Sie können API Gateway Version 2-APIs verwenden, um regionale benutzerdefinierte Domänennamen für REST-APIs und HTTP-APIs zu erstellen und zu verwalten. Für HTTP-APIs ist TLS 1.2 die einzige unterstützte TLS-Version.

Registrieren eines Domänennamens

Wenn Sie benutzerdefinierte Domänennamen für Ihre APIs einrichten möchten, müssen Sie eine Internetdomäne registrieren. Ihr Domainname muss der RFC 1035-Spezifikation entsprechen und darf ein Maximum von 63 Cents pro Label und 255 Cents insgesamt haben. Falls erforderlich, können Sie eine Internetdomäne mit Amazon Route 53 oder mit einem Domain-Registrar Ihrer Wahl registrieren. Der benutzerdefinierte Domänenname einer API kann dem Namen einer Unter- oder Stammdomäne (auch als „Zone Apex“ bezeichnet) einer registrierten Internetdomäne entsprechen.

Nachdem ein benutzerdefinierter Domänenname in API Gateway erstellt wurde, müssen Sie den Ressourceneintrag Ihres DNS-Providers erstellen oder aktualisieren, um ihn Ihrem API-Endpunkt zuzuordnen. Ohne eine solche Zuordnung können API-Anfragen für den benutzerdefinierten Domänennamen API Gateway nicht erreichen.

Regionale benutzerdefinierte Domänennamen

Wenn Sie einen benutzerdefinierten Domänennamen für eine regionale API erstellen, erstellt API Gateway einen regionalen Domänennamen für die API. Sie müssen einen DNS-Datensatz so einrichten, dass der benutzerdefinierte Domänenname dem regionalen Domänennamen zugeordnet wird. Außerdem müssen Sie ein Zertifikat für den benutzerdefinierten Domänennamen bereitstellen.

Benutzerdefinierte Domänennamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domänennamen können Sie eine nahezu unendliche Anzahl von Domänennamen unterstützen, ohne das Standardkontingentzu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domänennamen geben, *`customername`*`.api.example.com`.

Um einen benutzerdefinierten Platzhalterdomänennamen zu erstellen, geben Sie einen Platzhalter (*) als erste Subdomäne einer benutzerdefinierten Domäne an, die alle möglichen Subdomänen einer Root-Domäne darstellt.

Der benutzerdefinierte Domänenname mit Platzhalter `*.example.com` führt beispielsweise zu Unterdomänen wie `a.example.com`, `b.example.com` und `c.example.com`, die alle zur gleichen Domäne weiterleiten.

Benutzerdefinierte Domänennamen mit Platzhaltern unterstützen andere Konfigurationen als die benutzerdefinierten Standarddomänennamen von API Gateway. In einem einzigen AWS Konto können Sie beispielsweise unterschiedliche Einstellungen vornehmen `*.example.com` und `a.example.com` sich unterschiedlich verhalten.

Um einen benutzerdefinierten Domänennamen mit Platzhalter zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat angeben, das mithilfe der DNS- oder der E-Mail-Validierungsmethode validiert wurde.

> **Note**
>
> Sie können keinen benutzerdefinierten Platzhalter-Domänennamen erstellen, wenn ein anderes AWS Konto einen benutzerdefinierten Domänennamen erstellt hat, der mit dem benutzerdefinierten Platzhalter-Domänennamen in Konflikt steht. Wurde `a.example.com` beispielsweise von Konto A erstellt, dann kann Konto B als benutzerdefinierten Domänennamen mit Platzhalter nicht `*.example.com` erstellen.

Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das AWS Supportcenter wenden, um eine Ausnahme anzufordern.

Zertifikate für benutzerdefinierte Domänennamen

Important

Sie geben das Zertifikat für Ihren benutzerdefinierten Domänennamen an. Wenn Ihre Anwendung Zertifikat-Pinning, manchmal auch SSL-Pinning genannt, verwendet, um ein ACM-Zertifikat anzuheften, kann die Anwendung nach der Verlängerung des Zertifikats möglicherweise keine Verbindung zu Ihrer Domain herstellen. AWS Weitere Informationen finden Sie unter Probleme beim Zertifikats-Pinning im AWS Certificate Manager - Benutzerhandbuch.

Um ein Zertifikat für einen benutzerdefinierten Domänennamen in einer Region bereitzustellen, in der ACM unterstützt wird, müssen Sie ein Zertifikat von ACM anfordern. Um ein Zertifikat für einen regionalen benutzerdefinierten Domänennamen in einer von ACM nicht unterstützten Region bereitzustellen, müssen Sie ein Zertifikat für API Gateway in dieser Region importieren.

Wenn Sie ein SSL-/TLS-Zertifikat importieren möchten, müssen Sie den PEM-formatierten Hauptteil des SSL-/TLS-Zertifikats, den persönlichen Schlüssel und die Zertifikatkette für den benutzerdefinierten Domänennamen bereitstellen. Ein in ACM gespeichertes Zertifikat wird durch seinen ARN angegeben. Um ein AWS verwaltetes Zertifikat für einen Domainnamen zu verwenden, verweisen Sie einfach auf dessen ARN.

Die Einrichtung und Verwendung eines benutzerdefinierten Domänennamens für eine API in ACM ist ganz einfach. Sie erstellen ein Zertifikat für den angegebenen Domänennamen (oder importieren ein Zertifikat), richten den Domänennamen in API Gateway mit dem ARN des von ACM bereitgestellten Zertifikats ein und ordnen einen Basispfad unter dem benutzerdefinierten Domänennamen einer Bereitstellungsphase der API zu. Wenn Sie Zertifikate verwenden, die von ACM ausgestellt werden, brauchen sich keine Gedanken über vertrauliche Zertifikatdetails wie private Schlüssel zu machen.

Weitere Informationen zum Einrichten eines benutzerdefinierten Domänennamens finden Sie unter Vorbereiten von Zertifikaten in AWS Certificate Manager und Einrichten eines regionalen benutzerdefinierten Domänennamens in API Gateway.

Arbeiten mit API-Zuweisungen für HTTP-APIs

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Nachdem Sie einen Domain-Namen erstellt und DNS-Einträge konfiguriert haben, verwenden Sie API-Mappings, um Datenverkehr über Ihren benutzerdefinierten Domain-Namen an Ihre APIs zu senden.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise die `production`-Phase einer API in `https://api.example.com/orders` abbilden.

Sie können HTTP-API- und REST-API--Stufen demselben benutzerdefinierten Domain-Namen zuweisen.

Bevor Sie ein API-Mapping erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

Weiterleiten von API-Anforderungen

Sie können API-Mappings mit mehreren Ebenen konfigurieren, z. B. `orders/v1/items` und `orders/v2/items`.

Bei API-Mappings mit mehreren Ebenen leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspfad hat. API Gateway berücksichtigt nur die für API-Mappings konfigurierten Pfade und keine API-Routen, um die aufzurufende API auszuwählen. Wenn kein Pfad mit der Anforderung übereinstimmt, sendet API Gateway die Anforderung an die API, die Sie dem leeren Pfad zugeordnet habe `(none)`.

Bei benutzerdefinierten Domain-Namen, die API-Mappings mit mehreren Ebenen verwenden, leitet API Gateway Anfragen an das API-Mapping weiter, die den längsten Übereinstimmungspräfix hat.

Betrachten Sie beispielsweise einen benutzerdefinierten Domain-Namen `https://api.example.com` mit den folgenden API-Mappings:

1. `(none)` API 1 zugewiesen.
2. `orders` API 2 zugewiesen.
3. `orders/v1/items` API 3 zugewiesen.
4. `orders/v2/items` API 4 zugewiesen.

5. `orders/v2/items/categories` API 5 zugewiesen.

Anfrage	Ausgewählte API	Erklärung
`https://api.example.com/orders`	`API 2`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v1/items`	`API 3`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v2/items`	`API 4`	Die Anforderung stimmt genau mit diesem API-Mapping überein.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat. Das `123` am Ende der Anforderung hat keinen Einfluss auf die Auswahl.
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspfad hat.
`https://api.example.com/customers`	`API 1`	API Gateway verwendet das leere Mapping als Catch-All.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway wählt das Mapping aus, das den längsten Übereinstimmungspräfix hat. Bei einem benutzerdefinierten Domain-Namen, der mit einstufigen Mappings konfiguriert

Anfrage	Ausgewählte API	Erklärung
		ist, z. B. nur `https://api.example.com/orders` und `https://api.example.com/`, würde API Gateway `API 1` auswählen, da es keinen passenden Pfad mit `ordersandmore` gibt.

Einschränkungen

- Bei einer API-Zuordnung müssen sich der benutzerdefinierte Domainname und die zugewiesenen APIs im selben AWS Konto befinden.
- API-Mappings dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: `$-_.+!*'()/`.
- Die maximale Länge für den Pfad in eines API-Mappings beträgt 300 Zeichen.
- Es können 200 API-Zuweisungen mit mehreren Ebenen für jeden Domainnamen vorhanden sein.
- Sie können HTTP-APIs nur einem regionalen benutzerdefinierten Domain-Namen mit der TLS 1.2-Sicherheitsrichtlinie zuordnen.
- Sie können WebSocket APIs nicht demselben benutzerdefinierten Domainnamen zuordnen wie eine HTTP-API oder REST-API.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

AWS Serverless Application Model Vorlagen, die alle Ressourcen erstellen, finden Sie beispielsweise unter Sessions With SAM on GitHub.

AWS Management Console

So erstellen Sie ein API-Mapping

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie benutzerdefinierte Domain-Namen aus.
3. Wählen Sie einen benutzerdefinierten Domain-Namen aus, den Sie bereits erstellt haben.
4. Wählen Sie API-Mappings aus.
5. Wählen Sie API-Zuordnungen konfigurieren aus.
6. Wählen Sie Neue Zuordnung hinzufügen aus.
7. Geben Sie eine API, eine Phase und optional einen Pfad ein.
8. Wählen Sie Save (Speichern) aus.

AWS CLI

Der folgende AWS CLI Befehl erstellt eine API-Zuordnung. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1/orders` an die angegebene API und Phase.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

Das folgende AWS CloudFormation Beispiel erstellt eine API-Zuordnung.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Standardendpunkt für eine HTTP-API deaktivieren

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domänennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt.

> **Note**
>
> Wenn Sie den Standardendpunkt deaktivieren, wirkt sich dies auf alle Stufen einer API aus.

Der folgende AWS CLI Befehl deaktiviert den Standardendpunkt für eine HTTP-API.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird, es sei denn, automatische Bereitstellungen sind aktiviert.

Der folgende AWS CLI Befehl erstellt eine Bereitstellung.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

HTTP-API schützen

API Gateway bietet eine Reihe von Möglichkeiten, Ihre API vor bestimmten Bedrohungen zu schützen, wie böswillige Benutzer oder Spitzen im Datenverkehr. Sie können Ihre API mithilfe von Strategien wie das Festlegen von Drosselungs-Targets und das Aktivieren von gegenseitiger TLS schützen. In diesem Abschnitt erfahren Sie, wie Sie diese Funktionen mit API Gateway aktivieren können.

Themen

- Anfragen an Ihre HTTP-API drosseln
- Gegenseitige TLS-Authentifizierung für eine HTTP-API konfigurieren

Anfragen an Ihre HTTP-API drosseln

Sie können Drosselungen für Ihre APIs konfigurieren, um sie davor zu schützen, von zu vielen Anfragen überfordert zu werden. Drosselungen werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere überprüft API Gateway die Rate und einen Burst von Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise `429 Too Many Requests` Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Target-Limits für individuelle API-Stufen oder -Routen festlegen, um die Gesamtleistung in allen APIs in Ihrem Konto zu verbessern.

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) für alle APIs innerhalb eines AWS -Kontos pro Region. Die Steigerung (also die maximale Bucket-Größe) wird über alle APIs innerhalb eines AWS -Kontos pro Region wird ebenfalls beschränkt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von `429 Too Many Requests`-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter Kontingente und wichtige Hinweise.

Grenzwerte pro Konto werden auf alle APIs in einem Konto in einer bestimmten Region angewendet. Das Ratenlimit auf Kontoebene kann auf Anforderung erhöht werden – höhere Limits sind mit APIs möglich, die kürzere Timeouts und kleinere Nutzlasten aufweisen. Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das AWS Supportcenter. Weitere Informationen finden Sie unter Kontingente und wichtige Hinweise. Beachten Sie, dass diese Grenzwerte nicht höher als die AWS Drosselungsgrenzen sein können.

Drosselung auf Routenebene

Sie können die Drosselung auf Routenebene festlegen, um die Anforderungs-Drosselungslimits auf Kontoebene für eine bestimmte Stufe oder für individuelle Routen in Ihrer API zu überschreiben. Die Standardgrenzen für die Routendrosselung können die Ratenlimits auf Kontoebene nicht überschreiten.

Sie können die Drosselung auf Routenebene konfigurieren, indem Sie die verwende AWS CLI. Mit dem folgenden Befehl wird die benutzerdefinierte Einschränkung für die angegebene Stufe und Route einer API konfiguriert.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Gegenseitige TLS-Authentifizierung für eine HTTP-API konfigurieren

Die gegenseitige TLS-Authentifizierung erfordert eine bidirektionale Authentifizierung zwischen dem Client und dem Server. Bei gegenseitiger TLS müssen Clients X.509-Zertifikate zur Prüfung ihrer Identität präsentieren, um auf Ihre API zuzugreifen. Gegenseitiges TLS ist eine allgemeine Anforderung für das Internet der Dinge (IoT) und business-to-business Anwendungen.

Sie können gegenseitiges TLS zusammen mit anderen Autorisierungs- und Authentifizierungsvorgängen verwenden, die API Gateway unterstützt. API Gateway leitet die von Kunden bereitgestellten Zertifikate an Lambda-Genehmiger und an Backend-Integrationen weiter.

> **Important**
>
> Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Clients nur mithilfe eines benutzerdefinierten Domänennamens mit gegenseitiger TLS auf Ihre API zugreifen können, deaktivieren Sie den `execute-api`-Standardendpunkt. Weitere Informationen hierzu finden Sie unter the section called “Deaktivieren des Standardendpunkts”.

Voraussetzungen für gegenseitige TLS

Um gegenseitige TLS zu konfigurieren, benötigen Sie:

- Einen benutzerdefinierten Domänennamen
- AWS Certificate Manager Für Ihren benutzerdefinierten Domainnamen ist mindestens ein Zertifikat konfiguriert
- Ein in Amazon S3 konfigurierter und hochgeladener Truststore

Benutzerdefinierte Domänennamen

Um die gegenseitige TLS-Authentifizierung für eine HTTP-API zu aktivieren, müssen Sie einen benutzerdefinierten Domänennamen für Ihre API konfigurieren. Sie können gegenseitige TLS für einen benutzerdefinierten Domänennamen aktivieren und dann den benutzerdefinierten Domänennamen für Clients bereitstellen. Um mithilfe eines benutzerdefinierten Domänennamens auf eine API zuzugreifen, für die gegenseitige TLS aktiviert ist, müssen Clients Zertifikate präsentieren, denen Sie in API-Anforderungen vertrauen. Sie finden mehr Informationen in the section called “Benutzerdefinierte Domänennamen”.

Verwendung AWS Certificate Manager ausgestellter Zertifikate

Sie können ein öffentlich vertrauenswürdiges Zertifikat direkt von ACM anfordern oder öffentliche oder selbstsignierte Zertifikate importieren. Gehen Sie zum Einrichten eines Zertifikats in ACM zu ACM. Wenn Sie ein Zertifikat importieren möchten, lesen Sie weiter im folgenden Abschnitt.

Verwenden eines importierten AWS Private Certificate Authority Zertifikats oder Zertifikats

Um ein in ACM importiertes Zertifikat oder ein Zertifikat von Mutual TLS verwenden AWS Private Certificate Authority zu können, benötigt API Gateway ein von ACM `ownershipVerificationCertificate` ausgestelltes. Dieses Eigentumszertifikat wird nur verwendet, um zu überprüfen, ob Sie über Berechtigungen zur Verwendung des Domänennamens verfügen. Es wird nicht für den TLS-Handshake verwendet. Wenn Sie noch keine `ownershipVerificationCertificate` haben, navigieren Sie zuhttps://console.aws.amazon.com/acm/, um eine einzurichten.

Sie müssen dieses Zertifikat für die gesamte Lebensdauer Ihres Domainnamens gültig halten. Wenn ein Zertifikat abläuft und die automatische Verlängerung fehlschlägt, werden alle Updates des Domänennamens gesperrt. Sie müssen die `ownershipVerificationCertificateArn` mit einem gültigen `ownershipVerificationCertificate` aktualisieren, bevor Sie weitere Änderungen vornehmen können. Die `ownershipVerificationCertificate` kann nicht als Serverzertifikat für eine andere gemeinsame TLS-Domäne in API Gateway verwendet werden. Wenn ein Zertifikat direkt wieder in ACM importiert wird, muss der Aussteller gleich bleiben.

Konfigurieren Ihres Truststore

Truststores sind Textdateien mit einer `.pem`-Dateierweiterung. Sie sind eine vertrauenswürdige Liste von Zertifikaten von Zertifizierungsstellen. Um gegenseitige TLS zu verwenden, erstellen Sie einen Truststore von X.509-Zertifikaten, denen Sie vertrauen, für den Zugriff auf Ihre API.

Sie müssen die vollständige Vertrauenskette vom ausstellenden CA-Zertifikat bis zum Stamm-CA-Zertifikat in Ihren Truststore aufnehmen. API Gateway akzeptiert Clientzertifikate, die von jeder Zertifizierungsstelle in der Vertrauenskette ausgestellt werden. Die Zertifikate können von öffentlichen oder privaten Zertifizierungsstellen stammen. Zertifikate können eine maximale Kettenlänge von vier haben. Sie können auch selbstsignierte Zertifikate bereitstellen. Die folgenden Hash-Algorithmen werden im Truststore unterstützt:

- SHA-256 oder stärker
- RSA-2048 oder stärker
- ECDSA-256 oder stärker

API Gateway validiert eine Reihe von Zertifikatseigenschaften. Sie können Lambda-Genehmiger verwenden, um zusätzliche Prüfungen durchzuführen, wenn ein Client eine API aufruft, einschließlich der Prüfung, ob ein Zertifikat widerrufen wurde. API Gateway validiert die folgenden Eigenschaften:

Validierung	Beschreibung
X.509-Syntax	Das Zertifikat muss die X.509-Syntaxanford erungen erfüllen.
Integrität	Der Inhalt des Zertifikats darf sich nicht von dem unterscheiden, was von der Zertifizi erungsstelle des Truststore signiert wurde.
Gültigkeit	Die Gültigkeitsdauer des Zertifikats muss aktuell sein.
Namensverkettung/Schlüsselverkettung	Die Verkettung der Namen und Antragssteller von Zertifikaten darf nicht unterbrochen sein. Zertifikate können eine maximale Kettenlänge von vier haben.

Hochladen des Vertrauensspeichers in einen Amazon-S3-Bucket in einer einzigen Datei.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Der folgende AWS CLI Befehl lädt `certificates.pem` in Ihren Amazon S3 S3-Bucket hoch.

```
aws s3 cp certificates.pem s3://bucket-name
```

Konfigurieren von gegenseitiger TLS für einen benutzerdefinierten Domänennamen

Um gegenseitiges TLS für eine HTTP-API zu konfigurieren, müssen Sie einen benutzerdefinierten regionalen Domänennamen und mindestens TLS-Version 1.2 für Ihre API verwenden. Weitere Informationen zum Erstellen und Konfigurieren eines benutzerdefinierten Domänennamens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

> **Note**
>
> Gegenseitiges TLS wird für private APIs nicht unterstützt.

Nachdem Sie Ihren Vertrauensspeicher in Amazon S3 hochgeladen haben, können Sie Ihren benutzerdefinierten Domänennamen für die Verwendung von gegenseitigem TLS konfigurieren. Fügen Sie Folgendes (Schrägstriche enthalten) in ein Terminal ein:

```
aws apigatewayv2 create-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Nachdem Sie den Domänennamen erstellt haben, müssen Sie DNS-Datensätze und Basepath-Mappings für API-Vorgänge konfigurieren. Weitere Informationen hierzu finden Sie unter Einrichten eines regionalen benutzerdefinierten Domänennamens in API Gateway.

Aufrufen einer API mithilfe eines benutzerdefinierten Domänennamens, der gegenseitige TLS erfordert

Um eine API mit aktivierter gegenseitiger TLS aufzurufen, müssen Clients in der API-Anforderung ein vertrauenswürdiges Zertifikat präsentieren. Wenn ein Client versucht, Ihre API aufzurufen, sucht API Gateway in Ihrem Truststore nach dem Aussteller des Clientzertifikats. Damit API Gateway mit der Anforderung fortfahren kann, müssen sich der Aussteller des Zertifikats und die vollständige Vertrauenskette bis zum Stamm-CA-Zertifikats in Ihrem Truststore befinden.

Mit dem folgenden `curl`-Beispielbefehl wird eine Anforderung an `api.example.com`, gesendet, die `my-cert.pem` in die Anforderung einschließt. `my-key.key` ist der private Schlüssel für das Zertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Ihre API wird nur aufgerufen, wenn Ihr Truststore dem Zertifikat vertraut. Die folgenden Bedingungen führen dazu, dass API Gateway den TLS-Handshake fehlschlägt und die Anforderung mit einem `403`-Statuscode verweigert. Wenn Ihr Zertifikat:

- nicht vertrauenswürdig ist
- abgelaufen ist
- keinen unterstützten Algorithmus verwendet

> **Note**
>
> API Gateway überprüft nicht, ob ein Zertifikat widerrufen wurde.

Aktualisieren Ihres Truststore

Um die Zertifikate in Ihrem Truststore zu aktualisieren, laden Sie ein neues Zertifikatspaket in Amazon S3 hoch. Anschließend können Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um das aktualisierte Zertifikat zu verwenden.

Verwenden Sie Amazon-S3-Versioning, um mehrere Versionen Ihres Vertrauensspeichers zu verwalten. Wenn Sie Ihren benutzerdefinierten Domänennamen aktualisieren, um eine neue Vertrauensspeicher-Version zu verwenden, gibt API Gateway Warnungen zurück, wenn Zertifikate ungültig sind.

API Gateway erzeugt Zertifikatswarnungen nur dann, wenn Sie Ihren Domänennamen aktualisieren. API Gateway benachrichtigt Sie nicht, wenn ein zuvor hochgeladenes Zertifikat abläuft.

Der folgende AWS CLI Befehl aktualisiert einen benutzerdefinierten Domainnamen, sodass er eine neue Truststore-Version verwendet.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreVersion='abcdef123'
```

Gegenseitige TLS deaktivieren

Um gegenseitige TLS für einen benutzerdefinierten Domänennamen zu deaktivieren, entfernen Sie den Truststore aus Ihrem benutzerdefinierten Domänennamen, wie im folgenden Befehl gezeigt.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=''
```

Fehlerbehebung bei Zertifikatwarnungen

Wenn Sie einen benutzerdefinierten Domänennamen mit gegenseitiger TLS erstellen, gibt API Gateway Warnungen zurück, wenn Zertifikate im Truststore ungültig sind. Dies kann auch auftreten, wenn ein benutzerdefinierter Domänenname aktualisiert wird, um einen neuen Truststore zu verwenden. Die Warnungen geben das Problem mit dem Zertifikat und den Betreff des Zertifikats an, das die Warnung erstellt hat. Gegenseitige TLS ist weiterhin für Ihre API aktiviert, aber einige Clients können möglicherweise nicht auf Ihre API zugreifen.

Um aufzudecken, welches Zertifikat die Warnung erzeugt hat, dekodieren Sie die Zertifikate in Ihrem Truststore. Sie können Tools wie `openssl` zum Dekodieren der Zertifikate und zur Identifizierung ihrer Subjekte verwenden.

Der folgende Befehl zeigt den Inhalt eines Zertifikats einschließlich seines Betreffs an:

```
openssl x509 -in certificate.crt -text -noout
```

Aktualisieren oder entfernen Sie die Zertifikate, die Warnungen erstellt haben, und laden Sie dann einen neuen Truststore in Amazon S3 hoch. Nachdem Sie einen neuen Truststore hochgeladen haben, aktualisieren Sie Ihren benutzerdefinierten Domänennamen, um den neuen Truststore zu verwenden.

Problembehandlung bei Domänennamenskonflikten

Der Fehler `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` bedeutet, dass mehrere Zertifizierungsstellen ein Zertifikat für diese Domäne ausgestellt haben. Für jeden Subjekt im Zertifikat kann es nur einen Aussteller in API Gateway für gegenseitige TLS-Domänen geben. Sie müssen alle Ihre Zertifikate für dieses Thema über einen einzigen Aussteller abrufen. Wenn das Problem mit einem Zertifikat besteht, das Sie nicht kontrollieren können, Sie jedoch den Besitz des Domänennamens nachweisen können, kontaktieren Sie AWS Support, um ein Ticket zu eröffnen.

Fehlerbehebung bei Statusmeldungen für Domänennamen

`PENDING_CERTIFICATE_REIMPORT`: Das bedeutet, dass Sie ein Zertifikat erneut in ACM importiert haben und die Validierung fehlgeschlagen ist, da das neue Zertifikat über ein SAN (alternativer Antragstellername) verfügt, das nicht von der `ownershipVerificationCertificate` abgedeckt ist oder der Betreff oder die SANs im Zertifikat decken den Domänennamen nicht ab. Möglicherweise ist etwas falsch konfiguriert oder ein ungültiges Zertifikat wurde importiert. Sie müssen ein gültiges Zertifikat erneut in ACM importieren. Weitere Informationen zur Validierung finden Sie unter Validierung des Domänenbesitzes.

`PENDING_OWNERSHIP_VERIFICATION`: Dies bedeutet, dass Ihr zuvor verifiziertes Zertifikat abgelaufen ist und ACM es nicht automatisch erneuern konnte. Sie müssen das Zertifikat erneuern oder ein neues Zertifikat anfordern. Weitere Informationen zur Zertifikatserneuerung finden Sie im Leitfaden ACMs Fehlerbehebung bei der Erneuerung verwalteter Zertifikate.

HTTP-API überwachen

Sie können CloudWatch Metriken und CloudWatch Logs verwenden, um HTTP-APIs zu überwachen. Durch die Kombination von Protokollen und Metriken können Sie Fehler protokollieren und die Leistung Ihrer API überwachen.

> **Note**
>
> API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:
>
> - 413 Request Entity Too Large-Fehler
> - Übermäßige 429 Too Many Requests-Fehler
> - Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
> - Fehler der Serie 500, die durch interne Ausfälle verursacht werden

Themen

Mit Metriken für HTTP-APIs arbeiten

Sie können die API-Ausführung überwachen CloudWatch, indem Sie die Rohdaten von API Gateway sammeln und in lesbare near-real-time Metriken umwandeln. Diese Statistiken werden für einen Zeitraum von 15 Monaten aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt wird. Standardmäßig werden API-Gateway-Metrikdaten automatisch innerhalb von CloudWatch einer Minute gesendet. Um Ihre Metriken zu überwachen, erstellen Sie ein CloudWatch Dashboard für Ihre API. Weitere Informationen zum Erstellen eines CloudWatch Dashboards finden Sie unter Erstellen eines CloudWatch Dashboards im CloudWatch Amazon-Benutzerhandbuch. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch? im CloudWatch Amazon-Benutzerhandbuch.

Die folgenden Metriken werden für HTTP-APIs unterstützt. Sie können auch detaillierte Metriken aktivieren, um Metriken auf Routenebene an Amazon zu schreiben. CloudWatch

Metrik	Beschreibung
4xx	Die Anzahl der erfassten clientseitigen Fehler innerhalb eines bestimmten Zeitraums.

Metrik	Beschreibung
5xx	Die Anzahl der erfassten serverseitigen Fehler innerhalb eines bestimmten Zeitraums
Anzahl	Die Gesamtzahl der API-Anfor derungen innerhalb eines bestimmten Zeitraums
IntegrationLatency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage an das Backend weiterleitet und zu dem Zeitpunkt, zu dem es eine Antwort vom Backend erhält.
Latency	Die Zeit zwischen dem Zeitpunkt, zu dem API Gateway eine Anfrage von einem Client empfängt, und dem Zeitpunkt, an dem es eine Antwort an den Client zurückgibt. Die Latenzzeit schließt die Integrationslatenzzeit und anderen API Gateway-O verhead ein.
DataProcessed	Die Menge der verarbeiteten Daten in Bytes.

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

Dimension	Beschreibung
ApiId	Filtert API Gateway-M etriken für eine API mit der angegebenen API-ID.
ApiId, Phase	Filtert API Gateway-Metriken für eine API-Stufe mit der angegebenen API-ID und Stufen-ID.
ApiId, Methode, Ressource, Phase	Filtert API-Gateway-Metrik en für eine API-Methode mit der angegebenen API-ID, Stufen-ID, Ressourcenpfad und Routen-ID. API Gateway sendet diese Metriken nur, wenn Sie detaillierte CloudWatch Metriken explizit aktiviert haben. Sie können dies tun, indem Sie die UpdateSta geAktion der API Gateway V2 REST-API aufrufen, auf die die `detailedMetricsEna bled` Eigenschaft aktualisi ert `true` werden soll. Alternativ können Sie den AWS CLI Befehl update-stage aufrufen, um die `DetailedMetricsEna bled` Eigenschaft auf zu aktualisieren. `true` Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen Gebühren belastet. Preisinfo

Dimension	Beschreibung
	rmationen finden Sie unter CloudWatchAmazon-Preise.

Protokollierung für eine HTTP-API konfigurieren

Sie können die Protokollierung aktivieren, um Protokolle in Logs zu CloudWatch schreiben. Sie können Protokollierungsvariablen verwenden, um den Inhalt Ihrer Protokolle anzupassen.

Um die Protokollierung für eine HTTP-API einzuschalten, müssen Sie Folgendes tun.

1. Stellen Sie sicher, dass Ihr -Benutzer über die erforderlichen Berechtigungen zum Einschalten der Protokollierung verfügt.
2. Erstellen Sie eine CloudWatch Protokollgruppe für Logs.
3. Geben Sie den ARN der CloudWatch Logs-Protokollgruppe für eine Phase Ihrer API an.

Berechtigungen zum Einschalten der Protokollierung

Um die Protokollierung für eine API einzuschalten, muss der -Benutzer über die folgenden Berechtigungen verfügen.

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:CreateLogGroup",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": "*"
        }
    ]
}
```

Eine Protokollgruppe erstellen und die Protokollierung für HTTP-APIs aktivieren

Sie können eine Protokollgruppe erstellen und die Zugriffsprotokollierung mit dem AWS Management Console oder dem aktivieren AWS CLI.

AWS Management Console

1. Erstellen Sie eine -Protokollgruppe.

   Informationen zum Erstellen einer Protokollgruppe mithilfe der Konsole finden Sie unter Erstellen einer Protokollgruppe im Amazon CloudWatch Logs-Benutzerhandbuch.
2. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
3. Wählen Sie eine HTTP-API.
4. Wählen Sie unter der Registerkarte Monitor (Überwachen) im Hauptnavigationsbereich die Option Logging (Protokollierung) aus.
5. Wählen Sie eine Phase aus, um die Protokollierung zu aktivieren, und wählen Sie Select (Auswählen) aus.
6. Wählen Sie Edit (Bearbeiten) aus, um die Zugriffsprotokollierung zu aktivieren.
7. Aktivieren Sie die Zugriffsprotokollierung, geben Sie ein CloudWatch Protokoll ein und wählen Sie ein Protokollformat aus.
8. Wählen Sie Speichern.

AWS CLI

Der folgende AWS CLI Befehl erstellt eine Protokollgruppe.

```
aws logs create-log-group --log-group-name my-log-group
```

Zum Einschalten der Protokollierung benötigen Sie den Amazon-Ressourcennamen (ARN) für Ihre Protokollgruppe. *Das ARN-Format ist arn:aws:logs: region: account-id:log-group:. log-group-name*

Der folgende AWS CLI Befehl aktiviert die Protokollierung für die Phase einer HTTP-API. `$default`

```
aws apigatewayv2 update-stage --api-id abcdef \
    --stage-name '$default' \
    --access-log-settings '{"DestinationArn": "arn:aws:logs:region:account-
id:log-group:log-group-name", "Format": "$context.identity.sourceIp - -
 [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\"
 $context.status $context.responseLength $context.requestId"}'
```

Beispielprotokollformate

Beispiele für einige gängige Zugriffsprotokollformate sind in der API Gateway-Konsole verfügbar und im Folgenden aufgeführt.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp - - [$context.requestTime] "$context.httpMethod
 $context.routeKey $context.protocol" $context.status $context.responseLength
 $context.requestId $context.extendedRequestId
```

- `JSON`:

```
{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
 "requestTime":"$context.requestTime",
 "httpMethod":"$context.httpMethod","routeKey":"$context.routeKey",
 "status":"$context.status","protocol":"$context.protocol",
 "responseLength":"$context.responseLength", "extendedRequestId":
 "$context.extendedRequestId" }
```

- XML:

```
<request id="$context.requestId"> <ip>$context.identity.sourceIp</ip> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <routeKey>$context.routeKey</routeKey> <status>$context.status</status> <protocol>
$context.protocol</protocol> <responseLength>$context.responseLength</responseLength>
 <extendedRequestId>$context.extendedRequestId</extendedRequestId> </request>
```

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

HTTP-API-Zugriffsprotokolle anpassen

Sie können die folgenden Variablen zur Anpassung der HTTP-API-Zugriffsprotokolle verwenden. Um mehr über Zugriffsprotokolle für HTTP-APIs zu erfahren, lesen Sie Protokollierung für eine HTTP-API konfigurieren.

Parameter	Beschreibung
`$context.accountId`	Die AWS Konto-ID des API-Besitzers.
`$context.apiId`	Die ID, die API Gateway Ihrer API zuweist.
`$context.authorizer.claims.`*`property`*	Eine Eigenschaft der Ansprüche, die vom JSON Web Token (JWT) zurückgegeben wurden, nachdem der Methodenaufrufer erfolgreich authentifiziert wurde, z. B. `$context.authorizer.claims.username` Weitere Informationen finden Sie unter Steuern des Zugriffs auf HTTP-APIs mit JWT-Genehmigern.

Parameter	Beschreibung
	Note Bei einem Aufruf von `$context.authorizer.claims` wird null (0) zurückgegeben.
`$context.authorizer.error`	Die von einem Genehmiger zurückgegebene Fehlermeldung.
`$context.authorizer.principalId`	Die ID des Prinzipalbenutzers, die ein Lambda-Genehmiger zurückgibt.
`$context.authorizer.`*`property`*	Der Wert des angegebenen Schlüssel-Wert-Paares der `context`-Zuordnung, die von einer Lambda-Genehmigerfunktion des API Gateways zurückgegeben wird. Angenommen, der Genehmiger gibt folgende `context`-Zuweisung zurück: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` Der Aufruf von `$context.authorizer.key` gibt die Zeichenfolge `"value"` zurück, der Aufruf von `$context.authorizer.numKey` gibt 1 zurück, und der Aufruf von `$context.authorizer.boolKey` gibt `true` zurück.
`$context.awsEndpointRequestId`	Die Anforderungs-ID des AWS Endpunkts aus dem OR-Header`x-amz-request-id` . `x-amzn-requestId`

Parameter	Beschreibung
`$context.awsEndpointRequestId2`	Die Anforderungs-ID des AWS Endpunkts aus dem `x-amz-id-2` Header.
`$context.customDomain.basePathMatched`	Der Pfad für ein API-Mapping, mit dem eine eingehende Anforderung übereinstimmte. Gilt, wenn ein Client einen benutzerdefinierten Domain-Namen für den Zugriff auf eine API verwendet. Wenn ein Client beispielsweise eine Anforderung an `https://api.example.com/v1/orders/1234` sendet und die Anforderung dem API-Mapping mit dem Pfad `v1/orders` übereinstimmt, lautet der Wert `v1/orders` . Weitere Informationen hierzu finden Sie unter the section called “API-Zuweisungen”.
`$context.dataProcessed`	Die Menge der verarbeiteten Daten in Bytes.
`$context.domainName`	Der zum Aufrufen der API verwendete vollständige Domänennamen. Dieser Wert sollte mit dem für den eingehenden `Host`-Header übereinstimmen.
`$context.domainPrefix`	Das erste Label der `$context.domainName` .
`$context.error.message`	Eine Zeichenfolge, die eine API Gateway-Fehlermeldung enthält.
`$context.error.messageString`	Die Wert von `$context.error.message` in Anführungszeichen, d. h. `"$context.error.message"` .

Parameter	Beschreibung
`$context.error.responseType`	Ein Typ von `GatewayResponse` . Weitere Informationen erhalten Sie unter the section called “Metriken” und the section called “Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen”.
`$context.extendedRequestId`	Äquivalent mit `$context.requestId` .
`$context.httpMethod`	Die verwendete HTTP-Methode. Gültige Werte sind: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` und `PUT`.
`$context.identity.accountId`	Die der Anfrage zugeordnete AWS Konto-ID. Wird für Routen unterstützt, die die IAM-Autor isierung verwenden.
`$context.identity.caller`	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.

Parameter	Beschreibung
`$context.identity.cognitoAuthenticationProvider`	Eine durch Komma getrennte Liste der Amazon Cognito-Authentifizierungsanbieter, die vom Aufrufer, der die Anfrage stellt, verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Weitere Informationen finden Sie unter Verbundidentitäten verwenden im Amazon Cognito-Entwicklerhandbuch.
`$context.identity.cognitoAuthenticationType`	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind `authenticated` für authentifizierte Identitäten und `unauthenticated` für nicht authentifizierte Identitäten.
`$context.identity.cognitoIdentityId`	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
`$context.identity.cognitoIdentityPoolId`	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Beschreibung
`$context.identity.principalOrgId`	Die AWS -Organisations-ID. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
`$context.identity.clientCert.clientCertPem`	Das PEM-codierte Clientzertifikat, das der Client während der gegenseitigen TLS-Authentifizierung präsentiert hat. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
`$context.identity.clientCert.subjectDN`	Der Distinguished Name des Zertifikatantragsstellers, den ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
`$context.identity.clientCert.issuerDN`	Der Distinguished Name des Ausstellers des Zertifikats, das ein Client präsentiert. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
`$context.identity.clientCert.serialNumber`	Die Seriennummer des Zertifikats. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
`$context.identity.clientCert.validity.notBefore`	Das Datum, vor dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.

Parameter	Beschreibung
`$context.identity.clientCert.validity.notAfter`	Das Datum, nach dem das Zertifikat ungültig ist. Vorhanden, wenn ein Client mithilfe eines benutzerdefinierten Domänennamens, für den gegenseitige TLS aktiviert ist, auf eine API zugreift.
`$context.identity.sourceIp`	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
`$context.identity.user`	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
`$context.identity.userAgent`	Die `User-Agent` -Kopfzeile des API-Aufrufers.
`$context.identity.userArn`	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentifizierung. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	Die von einer Integration zurückgegebene Fehlermeldung. Äquivalent mit `$context.integrationErrorMessage` .
`$context.integration.integrationStatus`	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.

Parameter	Beschreibung
`$context.integration.latency`	Die Integrationslatenz in Millisekunden. Äquivalent mit `$context.integrationLatency` .
`$context.integration.requestId`	Die AWS Anforderungs-ID des Endpunkts . Äquivalent mit `$context.awsEndpointRequestId` .
`$context.integration.status`	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird.
`$context.integrationErrorMessage`	Eine Zeichenfolge, die eine Integrationsfehlermeldung enthält.
`$context.integrationLatency`	Die Integrationslatenz in Millisekunden.
`$context.integrationStatus`	Für die Lambda-Proxyintegration stellt dieser Parameter den Statuscode dar, der von der Backend-Lambda-Funktion zurückgegeben wird AWS Lambda, nicht von der Backend-Funktion.
`$context.path`	Der Anforderungspfad. Beispiel, `/{stage}/root/child` .

Parameter	Beschreibung
`$context.protocol`	Das Anforderungsprotokoll ist z. B, `HTTP/1.1`. **Note** API-Gateway-APIs können HTTP/2-Anfragen akzeptieren, aber API Gateway sendet Anfragen mithilfe von HTTP/1.1 an Backend-Integrationen. Infolgedessen wird das Anforderungsprotokoll als HTTP/1.1 protokolliert, auch wenn ein Client eine Anfrage sendet, die HTTP/2 verwendet.
`$context.requestId`	Die ID, die API Gateway der API-Anfrage zuweist.
`$context.requestTime`	Die Anforderungszeit im CLF-Format (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Die Anforderungszeit im Epoch-Format.
`$context.responseLatency`	Die Antwortlatenz in Millisekunden.
`$context.responseLength`	Die Länge der Antwortnutzlast in Byte.
`$context.routeKey`	Der Routing-Schlüssel der API-Anfrage, z. B. `/pets`.
`$context.stage`	Die Bereitstellungsstufe der API-Anforderung (z. B. `beta` oder `prod`).
`$context.status`	Der Status der Methodenantwort.

Fehlerbehebung bei Problemen mit HTTP-APIs

Die folgenden Themen enthalten Ratschläge zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung von HTTP-APIs auftreten können.

Themen

Fehlerbehebung bei Problemen mit HTTP-API-Lambda-Integrationen

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung von AWS Lambda Integrationen mit HTTP-APIs auftreten können.

Problem: Meine API mit einer Lambda-Integration gibt zurüc **`{"message":"Internal Server Error"}`**

Um den internen Serverfehler zu beheben, fügen Sie die `$context.integrationErrorMessage` Protokollierungsvariable zur Ihrem Protokollformat hinzu und sehen Sie sich die Protokolle Ihrer HTTP-API an. Um dies zu erreichen, gehen Sie wie folgt vor:

So erstellen Sie eine Protokollgruppe mit dem AWS Management Console

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Protokollgruppen.
3. Wählen Sie Protokollgruppe erstellen.
4. Geben Sie einen Protokollgruppennamen ein und wählen Sie dann Erstellen.
5. Notieren Sie sich den Amazon-Ressourcennamen (ARN) für Ihre Protokollgruppe. *`Das ARN-Format ist arn:aws:logs: region: account-id:log-group:. log-group-name`* Sie benötigen den ARN der Protokollgruppe, um die Zugriffsprotokollierung für Ihre HTTP-API zu aktivieren.

So fügen Sie die **`$context.integrationErrorMessage`**-Protokollierungsvariable hinzu

1. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

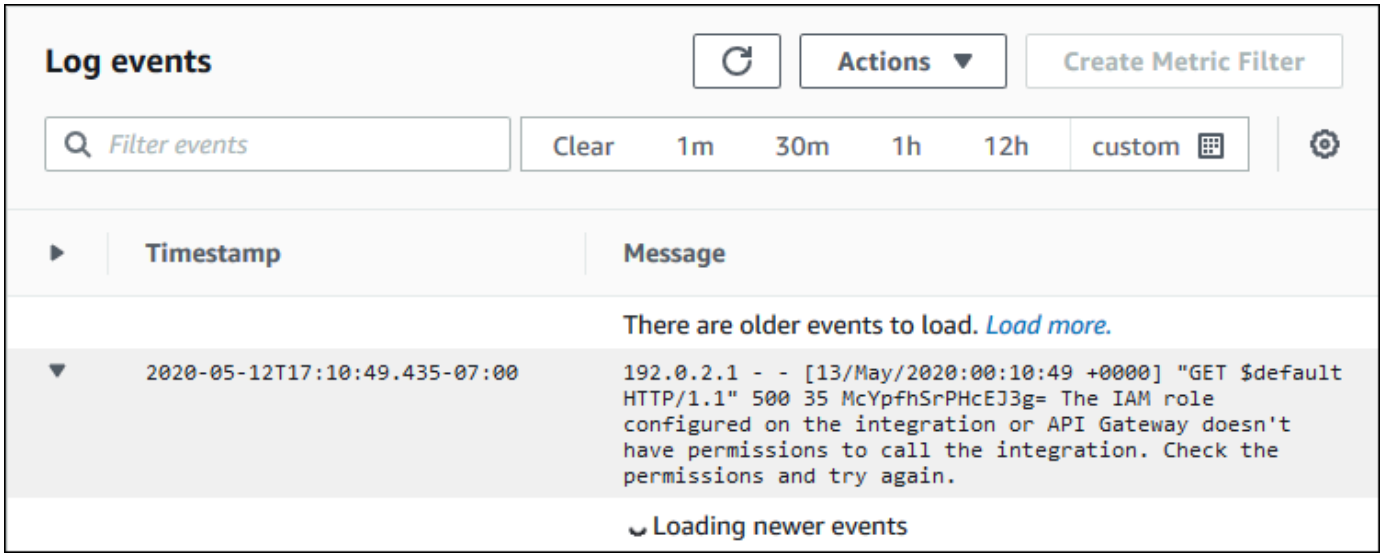
2. Wählen Sie Ihre HTTP-API.
3. Wählen Sie unter Monitor (Überwachen) die Option Logging (Protokollierung).
4. Wählen Sie eine Stufe Ihrer API aus.
5. Wählen Sie Bearbeiten und aktivieren Sie dann die Zugriffsprotokollierung.
6. Geben Sie für Protokollstandortden ARN der Protokollgruppe ein, die Sie im vorherigen Schritt erstellt haben.
7. Wählen Sie für Protokollformat die Option CLF aus. API Gateway erstellt ein Beispielprotokollformat.
8. Fügen Sie `$context.integrationErrorMessage` am Ende des Protokollformats hinzu.
9. Wählen Sie Save (Speichern) aus.

So zeigen Sie die Protokolle Ihrer API an

1. Generieren von Protokollen. Verwenden Sie einen Browser oder `curl`, um Ihre API aufzurufen.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Melden Sie sich bei der API Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
3. Wählen Sie Ihre HTTP-API.
4. Wählen Sie unter Monitor (Überwachen) die Option Logging (Protokollierung).
5. Wählen Sie die Stufe Ihrer API aus, für die Sie die Protokollierung aktiviert haben.
6. Wählen Sie „Logs in anzeigen CloudWatch“.
7. Wählen Sie den neuesten Protokoll-Stream, um die Protokolle Ihrer HTTP-API einzusehen.
8. Ihr Protokolleintrag sollte wie folgt aussehen:



Da wir `$context.integrationErrorMessage` dem Protokollformat hinzugefügt haben, wird in unseren Protokollen eine Fehlermeldung angezeigt, die das Problem zusammenfasst.

Ihre Protokolle enthalten möglicherweise eine andere Fehlermeldung, die darauf hinweist, dass ein Problem mit Ihrem Lambda-Funktionscode vorliegt. Überprüfen Sie in diesem Fall Ihren Lambda-Funktionscode und vergewissern Sie sich, dass Ihre Lambda-Funktion eine Antwort im erforderlichen Format zurückgibt. Wenn Ihre Protokolle keine Fehlermeldung enthalten, fügen Sie `$context.error.message` und `$context.error.responseType` Ihrem Protokollformat hinzu, um weitere Informationen zur Fehlerbehebung zu erhalten.

In diesem Fall zeigen die Protokolle, dass API Gateway nicht über die erforderlichen Berechtigungen zum Aufrufen der Lambda-Funktion verfügte.

Wenn Sie eine Lambda-Integration in der API Gateway-Konsole erstellen, konfiguriert API Gateway automatisch die Berechtigungen zum Aufrufen der Lambda-Funktion. Wenn Sie eine Lambda-Integration mithilfe des AWS CLI AWS CloudFormation, oder eines SDK erstellen, müssen Sie API Gateway Berechtigungen zum Aufrufen der Funktion gewähren. Die folgenden AWS CLI Beispielbefehle gewähren verschiedenen HTTP-API-Routen die Erlaubnis, eine Lambda-Funktion aufzurufen.

Example Beispiel — Für die **`$default`** Phase und die **`$default`** Route einer HTTP-API

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
```

```
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/\$default/\
$default"
```

Example Beispiel — Für die **`prod`** Phase und die **`test`** Route einer HTTP-API

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/prod/*/test"
```

Bestätigen Sie die Funktionsrichtlinie auf der Registerkarte Permissions (Berechtigungen) der Lambda-Konsole.

Versuchen Sie erneut, Ihre API aufzurufen. Sie sollten die Antwort Ihrer Lambda-Funktion sehen.

Fehlerbehebung bei Problemen mit HTTP-API JWT-Genehmigern

Im Folgenden finden Sie Hinweise zur Fehlerbehebung bei Fehlern und Problemen, die bei der Verwendung von JSON Web Token(JWT)-Genehmigern mit HTTP-APIs auftreten können.

Problem: Meine API gibt zurück **`401 {"message":"Unauthorized"}`**

Überprüfen Sie den `www-authenticate`-Header in der Antwort von der API.

Der folgende Befehl verwendet `curl`, um eine Anforderung an eine API mit einem JWT-Genehmiger zu senden, der `$request.header.Authorization` als Identitätsquelle verwendet.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-
west-2.amazonaws.com/route
```

Die Antwort von der API enthält einen `www-authenticate`-Header.

```
...
< HTTP/1.1 401 Unauthorized
< Date: Wed, 13 May 2020 04:07:30 GMT
< Content-Length: 26
< Connection: keep-alive
```

```
< www-authenticate: Bearer scope="" error="invalid_token" error_description="the token
 does not have a valid audience"
< apigw-requestid: Mc7UVioPPHcEKPA=
<
* Connection #0 to host api-id.execute-api.us-west-2.amazonaws.com left intact
{"message":"Unauthorized"}}
```

In diesem Fall zeigt der `www-authenticate`-Header an, dass das Token nicht für eine gültige Zielgruppe ausgegeben wurde. Damit API Gateway eine Anfrage autorisieren kann, muss der `aud`- oder `client_id`-Claim von JWT mit einem der Audience-Einträge übereinstimmen, die für den Genehmiger konfiguriert sind. API Gateway validiert `client_id` nur, wenn `aud` es nicht vorhanden ist. Wenn beide `aud` und vorhanden `client_id` sind, bewertet `aud` API Gateway.

Sie können auch ein JWT dekodieren und überprüfen, ob es mit dem Aussteller, der Zielgruppe und den Bereichen übereinstimmt, die Ihre API benötigt. Die Website jwt.io kann JWTs im Browser debuggen. Die OpenID Foundation pflegt eine Liste mit Bibliotheken für die Arbeit mit JWTs.

Weitere Informationen zu JWT-Genehmigern finden Sie unter Steuern des Zugriffs auf HTTP-APIs mit JWT-Genehmigern.

Mit WebSocket APIs arbeiten

Eine WebSocket API in API Gateway ist eine Sammlung von WebSocket Routen, die in Backend-HTTP-Endpunkte, Lambda-Funktionen oder andere Dienste integriert sind. AWS Sie können API Gateway-Funktionen verwenden, um Ihnen bei allen Aspekten des API-Lebenszyklus zu helfen, von der Erstellung bis zur Überwachung Ihrer Produktions-APIs.

WebSocket API-Gateway-APIs sind bidirektional. Ein Client kann Nachrichten an einen Service senden und Services können unabhängig Nachrichten an Clients senden. Dieses bidirektionale Verhalten ermöglicht umfassendere Interaktionen zwischen Client und Service, da Dienste Daten an Clients weiterleiten können, ohne dass Kunden eine ausdrückliche Anfrage stellen müssen. WebSocket APIs werden häufig in Echtzeitanwendungen wie Chat-Anwendungen, Kollaborationsplattformen, Multiplayer-Spielen und Finanzhandelsplattformen verwendet.

Eine Beispiel-App, mit der Sie beginnen können, finden Sie unter Tutorial: Eine serverlose Chat-App mit einer WebSocket API, Lambda und DynamoDB erstellen.

In diesem Abschnitt erfahren Sie, wie Sie Ihre WebSocket APIs mithilfe von API Gateway entwickeln, veröffentlichen, schützen und überwachen.

Themen

Über WebSocket APIs in API Gateway

In API Gateway können Sie eine WebSocket API als Stateful-Frontend für einen AWS Dienst (wie Lambda oder DynamoDB) oder für einen HTTP-Endpunkt erstellen. Die WebSocket API ruft Ihr Backend auf der Grundlage des Inhalts der Nachrichten auf, die sie von Client-Apps empfängt.

Im Gegensatz zu einer REST-API, die Anfragen empfängt und beantwortet, unterstützt eine WebSocket API die bidirektionale Kommunikation zwischen Client-Apps und Ihrem Backend. Das Backend kann Rückruf-Nachrichten an verbundene Clients senden.

In Ihrer WebSocket API werden eingehende JSON-Nachrichten auf der Grundlage von Routen, die Sie konfigurieren, an Backend-Integrationen weitergeleitet. (Nicht-JSON-Nachrichten werden an eine von Ihnen konfigurierte `$default`-Route weitergeleitet.)

Eine Route umfasst einen Routenschlüssel. Dabei handelt es sich um den Wert, der bei der Auswertung eines Routen-Auswahlausdrucks erwartet wird. Das `routeSelectionExpression` ist ein auf API-Ebene definiertes Attribut. Es gibt eine JSON-Eigenschaft an, von der erwartet wird, dass sie in der Nachricht-Nutzlast vorhanden ist. Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter the section called “”.

Beispiel: Wenn Ihre JSON-Nachrichten eine `action`-Eigenschaft enthalten und Sie basierend auf dieser Eigenschaft verschiedene Aktionen durchführen möchten, könnte Ihr Routen-Auswahlausdruck `${request.body.action}` lauten. Ihre Routing-Tabelle würde in diesem Fall angeben, welche Aktion ausgeführt werden soll, indem der Wert der `action`-Eigenschaft gegen die benutzerdefinierten Routenschlüssel-Werte abgeglichen wird, die Sie in der Tabelle definiert haben.

Es gibt drei vordefinierte Routen, die verwendet werden können: `$connect`, `$disconnect` und `$default`. Darüber hinaus können Sie benutzerdefinierte Routen erstellen.

- API Gateway ruft die `$connect` Route auf, wenn eine persistente Verbindung zwischen dem Client und einer WebSocket API initiiert wird.
- API Gateway ruft die `$disconnect`-Route auf, wenn der Client oder der Server die Verbindung mit der API unterbricht.
- API Gateway ruft eine benutzerdefinierte Route auf, wenn nach der Auswertung des Routen-Auswahlausdrucks im Hinblick auf die Nachricht eine übereinstimmende Route gefunden wird. Die Übereinstimmung bestimmt, welche Integration aufgerufen wird.
- API Gateway ruft die Route `$default` auf, wenn der Routen-Auswahlausdruck nicht im Hinblick auf die Nachricht ausgewertet werden kann oder wenn keine übereinstimmende Route gefunden wird.

Weitere Informationen über die Routen `$connect` und `$disconnect` finden Sie unter the section called “Verwalten von verbundenen Benutzern und Client-Apps”.

Weitere Informationen über die Route `$default` und benutzerdefinierte Routen finden Sie unter the section called “Aufrufen Ihrer Backend-Integration”.

Backend-Services können Daten an verbundene Client-Apps senden. Weitere Informationen finden Sie unter the section called “Senden von Daten von Backend-Services an verbundene Clients”.

Verwalten von verbundenen Benutzern und Client-Apps: Routen **$connect** und **$disconnect**

Themen

Die Route **$connect**

Client-Apps stellen eine Verbindung zu Ihrer WebSocket API her, indem sie eine WebSocket Upgrade-Anfrage senden. Wenn die Anforderung erfolgreich ist, wird die Route `$connect` ausgeführt, während die Verbindung hergestellt wird.

Da es sich bei der WebSocket Verbindung um eine statusbehaftete Verbindung handelt, können Sie die Autorisierung nur für die `$connect` Route konfigurieren. `AuthN/AuthZ`wird nur zur Verbindungszeit ausgeführt.

Solange die Integration im Zusammenhang mit der Route `$connect` noch nicht abgeschlossen wurde, steht die Upgrade-Anforderung noch an und die tatsächliche Verbindung wird nicht hergestellt. Wenn die `$connect`-Anforderung fehlschlägt (z. B. aufgrund eines `AuthN/AuthZ`-Fehlers oder eines Integrationsfehlers), wird die Verbindung nicht hergestellt.

> **Note**
>
> Wenn die Autorisierung für `$connect` fehlschlägt, wird keine Verbindung hergestellt, und der Client erhält eine `401`- oder `403`-Antwort.

Das Einrichten einer Integration für `$connect` ist optional. Sie sollten die Einrichtung einer `$connect`-Integration unter folgenden Umständen erwägen:

- Sie möchten Clients ermöglichen, Unterprotokolle mithilfe des `Sec-WebSocket-Protocol`-Felds anzugeben. Beispielcode finden Sie unter Einrichtung einer `$connect` Route, für die ein WebSocket Unterprotokoll erforderlich ist.
- Sie möchten benachrichtigt werden, wenn Clients verbunden werden.

- Wenn Sie Verbindungen drosseln möchten oder wenn Sie steuern möchten, wer eine Verbindung herstellt.
- Wenn Sie möchten, dass Ihr Backend über eine Rückruf-URL Nachrichten an Clients zurücksendet.
- Sie möchten jede Verbindungs-ID und andere Informationen in einer Datenbank (z. B. Amazon DynamoDB) speichern.

Verbindungsinformationen von der **`$connect`**-Route übergeben

Sie können sowohl Proxy- als auch Nicht-Proxy-Integrationen verwenden, um Informationen von der `$connect`-Route an eine Datenbank oder einen anderen AWS-Service zu übergeben.

So übergeben Sie Verbindungsinformationen mithilfe einer Proxy-Integration

In diesem Fall können Sie über eine Lambda-Proxy-Integration auf die Verbindungsinformationen zugreifen. Verwenden Sie eine andere AWS-Service AWS Lambda OR-Funktion, um Beiträge an die Verbindung zu senden.

Die folgende Lambda-Funktion zeigt, wie das `requestContext`-Objekt verwendet wird, um die Verbindungs-ID, den Domänennamen, den Stufennamen und die Abfragezeichenfolgen zu protokollieren.

Node.js

```
 export const handler = async(event, context) => {
    const connectId = event["requestContext"]["connectionId"]
    const domainName = event["requestContext"]["domainName"]
    const stageName = event["requestContext"]["stage"]
    const qs = event['queryStringParameters']
    console.log('Connection ID: ', connectId, 'Domain Name: ', domainName, 'Stage
 Name: ', stageName, 'Query Strings: ', qs )
    return {"statusCode" : 200}
};
```

Python

```
import json
import logging
logger = logging.getLogger()
logger.setLevel("INFO")
```

```
def lambda_handler(event, context):
    connectId = event["requestContext"]["connectionId"]
    domainName = event["requestContext"]["domainName"]
    stageName = event["requestContext"]["stage"]
    qs = event['queryStringParameters']
    connectionInfo = {
        'Connection ID': connectId,
        'Domain Name': domainName,
        'Stage Name': stageName,
        'Query Strings': qs}
    logging.info(connectionInfo)
    return {"statusCode": 200}
```

So übergeben Sie Verbindungsinformationen mithilfe einer Nicht-Proxy-Integration

- Sie können mit einer Nicht-Proxy-Integration auf die Verbindungsinformationen zugreifen. Richten Sie die Integrationsanfrage ein und stellen Sie eine WebSocket API-Anforderungsvorlage bereit. Die folgende Zuordnungsvorlage der Velocity Template Language (VTL) enthält eine Integrationsanforderung. Diese Anforderung sendet die folgenden Details an eine Nicht-Proxy-Integration:

  - Verbindungs-ID
  - Domainname
  - Stufenname
  - Pfad
  - Überschriften
  - Abfragezeichenfolgen

  Diese Anforderung sendet die Verbindungs-ID, den Domain-Namen, den Namen der Phase, die Pfade, Header und Abfragezeichenfolgen an eine Nicht-Proxy-Integration.

```
{
    "connectionId": "$context.connectionId",
    "domain": "$context.domainName",
    "stage": "$context.stage",
    "params": "$input.params()"
```

```
}
```

Weitere Informationen zum Einrichten von Datenübertragungen finden Sie unterthe section called “Datentransformationen”.

Zum Abschließen der Integrationsanforderung legen Sie die Integrationsantwort `StatusCode: 200` fest. Weitere Informationen zum Einrichten einer Integrationsantwort finden Sie unter Integrationsantwort mit der API Gateway-Konsole einrichten.

Die Route **`$disconnect`**

Die Route `$disconnect` wird ausgeführt, nachdem die Verbindung geschlossen wurde.

Die Verbindung kann vom Server oder vom Client geschlossen werden. Da die Verbindung bereits geschlossen ist, wenn sie ausgeführt wird, ist `$disconnect` ein "Best Effort"-Ereignis. API Gateway wird sich nach bemühen, das `$disconnect`-Ereignis an Ihre Integration zu übermitteln, kann aber die Übermittlung nicht garantieren.

Das Backend kann die Verbindungstrennung mithilfe der `@connections`-API einleiten. Weitere Informationen finden Sie unter the section called “Verwenden der `@connections`-Befehle in Ihrem Backend-Service”.

Aufrufen Ihrer Backend-Integration: **`$default`**-Route und benutzerdefinierte Routen

Themen

- Verwenden von Routen zur Verarbeitung von Nachrichten
- Die Route $default
- Benutzerdefinierte Routen
- Verwenden Sie API Gateway WebSocket API-Integrationen, um eine Verbindung zu Ihrer Geschäftslogik herzustellen
- Wichtige Unterschiede zwischen WebSocket APIs und REST-APIs

Verwenden von Routen zur Verarbeitung von Nachrichten

In API Gateway WebSocket Gateway-APIs können Nachrichten vom Client an Ihren Backend-Service gesendet werden und umgekehrt. Im Gegensatz zum Anforderungs-/Antwortmodell von HTTP

können WebSocket im Backend Nachrichten an den Client gesendet werden, ohne dass der Client Maßnahmen ergreift.

Es kann sich dabei um JSON- oder Nicht-JSON-Nachrichten handeln. Es können jedoch nur JSON-Nachrichten basierend auf dem Nachrichteninhalt an bestimmte Integrationen weitergeleitet werden. Nicht-JSON-Nachrichten werden über die `$default`-Route an das Backend übergeben.

> **Note**
>
> API Gateway unterstützt Nachrichten-Payloads bis zu 128 KB mit einer maximalen Frame-Größe von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind. Wenn eine größere Nachricht (oder ein größerer Frame) empfangen wird, wird die Verbindung mit dem Code 1009 geschlossen.
> Derzeit werden keine binären Nutzlasten unterstützt. Wenn ein binärer Frame empfangen wird, wird die Verbindung mit dem Code 1003 geschlossen. Es ist jedoch möglich, binäre Nutzlasten in Text zu konvertieren. Siehe the section called “Binäre Medientypen”.

Mit WebSocket APIs in API Gateway können JSON-Nachrichten weitergeleitet werden, um einen bestimmten Backend-Service auf der Grundlage des Nachrichteninhalts auszuführen. Wenn ein Client eine Nachricht über seine WebSocket Verbindung sendet, führt dies zu einer Routing-Anfrage an die WebSocket API. Die Anfrage wird der Route mit dem entsprechenden Routenschlüssel in API Gateway zugeordnet. Sie können eine Routenanforderung für eine WebSocket API in der API Gateway Gateway-Konsole einrichten AWS CLI, indem Sie das oder ein AWS SDK verwenden.

> **Note**
>
> In den AWS SDKs AWS CLI und können Sie Routen erstellen, bevor oder nachdem Sie Integrationen erstellt haben. Derzeit unterstützt die Konsole keine Wiederverwendung von Integrationen. Daher müssen Sie zuerst die Route erstellen und dann die Integration für diese Route.

Sie können API Gateway so konfigurieren, dass die Validierung einer Routenanforderung durchgeführt wird, bevor Sie mit der Integrationsanforderung fortfahren. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage fehl, ohne Ihr Backend aufzurufen, sendet eine `"Bad request body"` Gateway-Antwort ähnlich der folgenden an den Client und veröffentlicht die Validierungsergebnisse in CloudWatch Logs:

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId":
 "{messageId}"}
```

Damit werden unnötige Aufrufe Ihres Backend reduziert, sodass Sie sich auf andere Voraussetzungen für Ihre API konzentrieren können.

Darüber hinaus können Sie eine Routenantwort für die Routen Ihrer API vorgeben, um eine bidirektionale Kommunikation zu ermöglichen. Eine Routenantwort beschreibt, welche Daten bei Abschluss der Integration einer bestimmten Route an Ihren Client gesendet werden. Es ist nicht notwendig, eine Antwort für eine Route vorzugeben, wenn ein Client Nachrichten an Ihr Backend senden soll, ohne eine Antwort zu erhalten (unidirektionale Kommunikation). Wenn Sie keine Routenantwort bereitstellen, sendet API Gateway jedoch keine Informationen über das Ergebnis Ihrer Integration an Ihre Kunden.

Die Route `$default`

Jede WebSocket API-Gateway-API kann eine `$default` Route haben. Hierbei handelt es sich um einen speziellen Routing-Wert, der auf folgende Weise eingesetzt werden kann:

- Sie können ihn zusammen mit vorgegebenen Routenschlüsseln verwenden, um für eingehende Nachrichten, auf die keine der vorgegebenen Routenschlüssel zutreffen, eine "Fallback"-Route anzugeben (z. B. eine generische Pseudo-Integration, die eine bestimmte Fehlermeldung zurückgibt).
- Sie können ihn ohne vorgegebene Routenschlüssel verwenden, um ein Proxy-Modell anzugeben, durch das Routing an eine Backend-Komponente delegiert wird.
- Sie können mit ihm eine Route für Nicht-JSON-Nutzlasten angeben.

Benutzerdefinierte Routen

Wenn basierend auf dem Nachrichteninhalt eine bestimmte Integration aufgerufen werden soll, können Sie hierzu eine benutzerdefinierte Route erstellen.

Für eine benutzerdefinierte Route werden ein Routenschlüssel und eine Integration nach Ihren Angaben verwendet. Wenn eine eingehende Nachricht eine JSON-Eigenschaft enthält und diese Eigenschaft mit einem Wert ausgewertet wird, der mit dem Routenschlüsselwert übereinstimmt, ruft API Gateway die Integration auf. (Weitere Informationen finden Sie unter the section called “Über WebSocket APIs”.)

Angenommen, Sie möchten eine Chat-Raum-Anwendung erstellen. Sie könnten damit beginnen, eine WebSocket API zu erstellen, deren Routenauswahlausdruck lautet`$request.body.action`. Anschließend könnten Sie zwei Routen definieren: `joinroom` und `sendmessage`. Ein Client-App könnte die `joinroom`-Route durch Senden einer Nachricht wie die folgende aufrufen:

```
{"action":"joinroom","roomname":"developers"}
```

Und sie könnte die `sendmessage`-Route durch Senden einer Nachricht wie die folgende aufrufen:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Verwenden Sie API Gateway WebSocket API-Integrationen, um eine Verbindung zu Ihrer Geschäftslogik herzustellen

Nachdem Sie eine Route für eine WebSocket API-Gateway-API eingerichtet haben, müssen Sie die Integration angeben, die Sie verwenden möchten. Genauso wie eine Route über eine Routenanforderung und eine Routenantwort verfügen kann, kann eine Integration eine Integrationsanforderung und eine Integrationsantwort besitzen. Eine Integrationsanforderung enthält die Informationen, die von Ihrem Backend zur Verarbeitung der von Ihrem Client stammenden Anforderung erwartet werden. Die Integrationsantwort enthält die Daten, die Ihr Backend an API Gateway zurücksendet. Sie kann verwendet werden, um eine an den Client zu sendende Nachricht zu verfassen (falls eine Routenantwort definiert ist).

Weitere Informationen zum Einrichten von Integrationen finden Sie unter the section called “Integrationen”.

Wichtige Unterschiede zwischen WebSocket APIs und REST-APIs

Integrationen für WebSocket APIs ähneln Integrationen für REST-APIs, mit Ausnahme der folgenden Unterschiede:

- Derzeit müssen Sie in der API Gateway-Konsole zuerst eine Route und dann eine Integration als Ziel dieser Route erstellen. In der API und der CLI können Sie Routen und Integrationen jedoch unabhängig voneinander in beliebiger Reihenfolge erstellen.
- Sie können eine einzelne Integration für mehrere Routen verwenden. Für eine Gruppe von Aktionen, die eng miteinander in Beziehung stehen, empfiehlt es sich beispielsweise, dass alle diese Routen zu einer einzelnen Lambda-Funktion führen. Anstatt die Details der Integration mehrmals zu definieren, können Sie sie einmal angeben und jeder verwandten Route zuweisen.

> **Note**
>
> Derzeit unterstützt die Konsole keine Wiederverwendung von Integrationen. Daher müssen Sie zuerst die Route erstellen und dann die Integration für diese Route.
> In den AWS SDKs AWS CLI und können Sie eine Integration wiederverwenden, indem Sie das Ziel der Route auf einen Wert von setzen"`integrations/{integration-id}`", wobei die eindeutige ID der Integration "`{integration-id}`" steht, die der Route zugeordnet werden soll.

- API Gateway bietet mehrere Auswahlausdrücke, die Sie in Ihren Routen und Integrationen verwenden können. Sie müssen die Auswahl einer Eingabevorlage oder einer Ausgabezuordnung nicht vom Inhaltstyp abhängig machen. Wie bei Routenauswahlausdrücken können Sie einen Auswahlausdruck definieren, der von API Gateway ausgewertet wird, um das richtige Element auszuwählen. Alle von ihnen übernehmen automatisch die `$default`-Vorlage, wenn keine passende Vorlage gefunden wird.
  - In Integrationsanforderungen unterstützt der Vorlagen-Auswahlausdruck `$request.body.<json_path_expression>` und statische Werte.
  - In Integrationsantworten unterstützt der Vorlagen-Auswahlausdruck `$request.body.<json_path_expression>`, `$integration.response.statuscode` und `$integration.response.header.<headerName>` sowie statische Werte.

Im HTTP-Protokoll, in dem Anforderungen und Antworten synchron gesendet werden, ist die Kommunikation im Wesentlichen unidirektional. Im WebSocket Protokoll erfolgt die Kommunikation in beide Richtungen. Antworten sind asynchron und werden vom Client nicht unbedingt in der gleichen Reihenfolge empfangen, in der die Client-Nachrichten gesendet wurden. Darüber hinaus kann das Backend Nachrichten an den Client senden.

> **Note**
>
> Für eine Route, die zur Verwendung der Integration `AWS_PROXY` oder `LAMBDA_PROXY` konfiguriert wurde, ist die Kommunikation unidirektional. API Gateway übergibt die Backend-Antwort nicht automatisch über die Routenantwort. Wenn bei der `LAMBDA_PROXY`-Integration wird der von der Lambda-Funktion zurückgegebene Text beispielsweise nicht an den Client zurückgegeben. Wenn Sie möchten, dass der Client Integrationsantworten erhält, müssen Sie eine Routenantwort definieren, um eine bidirektionale Kommunikation zu ermöglichen.

Senden von Daten von Backend-Services an verbundene Clients

API Gateway WebSocket Gateway-APIs bieten Ihnen die folgenden Möglichkeiten, Daten von Back-End-Diensten an verbundene Clients zu senden:

- Eine Integration kann eine Antwort senden, die über die von Ihnen definierte Routenantwort an den Client zurückgegeben wird.
- Sie können mithilfe der `@connections`-API eine POST-Anforderung senden. Weitere Informationen finden Sie unter the section called “Verwenden der `@connections`-Befehle in Ihrem Backend-Service”.

WebSocket Auswahlausdrücke in API Gateway

Themen

API Gateway verwendet Auswahlausdrücke als Möglichkeit, den Anfrage- und Antwortkontext auszuwerten und einen Schlüssel zu erzeugen. Anhand des Schlüssels wird dann aus einem Satz möglicher Werte ausgewählt, die gewöhnlich von Ihnen, dem API-Entwickler, bereitgestellt werden. Der genaue Satz der unterstützten Variablen ist vom jeweiligen Ausdruck abhängig. Jeder Ausdruck wird nachstehend weiter erörtert.

Für alle Ausdrücke folgt die Sprache dem gleichen Satz von Regeln:

- Einer Variable wird vorangestell `"$"`.
- Variablen können explizit durch geschweifte Klammern abgegrenzt werden, z. ., `"${request.body.version}-beta"`.
- Mehrere Variablen werden unterstützt, aber die Auswertung findet nur einmal statt (keine rekursive Auswertung).
- Ein Dollarzeichen ($) kann durch die Escape-Zeichen `"\"` geschützt werden. Dies ist besonders nützlich, wenn ein Ausdruck definiert wird, der dem reservierten `$default`-Schlüssel zugeordnet wird, z. B. `"\$default"`.

- In einigen Fällen ist ein Musterformat erforderlich. In diesem Fall sollte der Ausdruck entsprechend `"/"`-Statuscodes mit Schrägstrichen (`"/2\d\d/"`) verpackt werden, z. B. 2*XX*.

Routenantwort-Auswahlausdrücke

Eine Routenantwort wird für die Modellierung einer Antwort vom Backend zum Client verwendet. Für WebSocket APIs ist eine Routenantwort optional. Wenn es definiert ist, signalisiert es API Gateway, dass es beim Empfang einer WebSocket Nachricht eine Antwort an einen Client zurückgeben soll.

Die Auswertung des Routenantwort-Auswahlausdrucks ergibt einen Routenantwort-Schlüssel. Zukünftig wird dieser Schlüssel zur Auswahl unter einer der mit der API verknüpften `RouteResponses` dienen. Zurzeit wird jedoch nur der Schlüssel `$default` unterstützt.

API-Schlüssel-Auswahlausdrücke

Dieser Ausdruck wird ausgewertet, wenn der Service bestimmt, dass mit der vorliegenden Anforderung nur fortgefahren werden soll, wenn der Client einen gültigen API-Schlüssel bereitstellt.

Zurzeit werden als einzigen Werte `$request.header.x-api-key` und `$context.authorizer.usageIdentifierKey` unterstützt.

API-Zuordnungs-Auswahlausdrücke

Dieser Ausdruck wird ausgewertet, um zu bestimmen, welche API-Stufe bei einer über eine benutzerdefinierte Domäne gestellten Anforderung auszuwählen ist.

Derzeit wird als einziger Wert unterstütz `$request.basepath`.

WebSocketZusammenfassung des Auswahlausdrucks

In der folgenden Tabelle werden die Anwendungsfälle für Auswahlausdrücke in WebSocket APIs zusammengefasst:

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`Api.RouteSelectionExpression`	`Route.RouteKey`	`$defaul` wird als	Leitet WebSocket Nachricht

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
		Catch-all-Route unterstützt.	en auf der Grundlage des Kontextes einer Client-Anfrage weiter.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`Route.ModelSelectionExpression`	Schlüssel für `Route.RequestModels`	Optional. Bei Angabe für Nicht-Proxy-Integrationen findet eine Modellvalidierung statt. `$defaul` wird als Catch-all-Route unterstützt.	Führt die Anforderungsvalidierung dynamisch innerhalb derselben Route aus.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`Integration.TemplateSelectionExpression`	Schlüssel für `Integration.RequestTemplates`	Optional. Kann für Nicht-Proxy-Integrationen zum Bearbeiten von eingehenden Nutzlasten bereitgestellt werden. `${request.body.jsonPath}` zte Werte sind und statische Werte. `$default` wird als Catch-	Dient zur Bearbeitung der Anforderung der aufrufenden Methode je nach den dynamischen Eigenschaften der Anforderung.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
		all-Route unterstützt.	

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`Integration.IntegrationResponseSelectionExpression`	`IntegrationResponse.IntegrationResponseKey`	Optional. Kann für Nicht-Proxy-Integrationen bereitgestellt werden. Dient als Musterve gleich für Fehlerme dungen (von Lambda) oder Statusco es (von HTTP-Inte grationer). `$defaul` ist erforderl ich,	Dient zum Bearbeiten der Antwort vom Backend. Wählen Sie, welche Aktion basierend auf der dynamischen Antwort des Backends ausgeführt werden soll (z. B. unterschiedliche Behandlung bestimmter Fehler).

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
		damit Nicht-Proxy-Integrationen als Catch-All für erfolgreiche Antworten fungieren.	

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`IntegrationResponse.TemplateSelectionExpression`	Schlüssel für `IntegrationResponse.ResponseTemplates`	Optional. Kann für Nicht-Proxy-Integrationen bereitgestellt werden. $default wird unterstützt.	In einigen Fällen kann eine dynamische Eigenschaft der Antwort verschiedene Transformationen innerhalb derselben Route und zugehörigen Integration vorgeben. `${request.body.jsonPath}` , `${integration.response.statuscode}` , `${integration.resp`

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
			`onse.header.headerName}` , `${integration.response.multivalueheader.headerName}` , Unterstützte Werte sind , , , und statische Werte. `$default` wird als Catch-all-Route unterstützt.

Auswahlausdrücke	Ausgewertet auf Schlüssel für	Hinweise	Beispielanwendungsfall
`Route.RouteResponseSelectionExpression`	`RouteResponse.RouteResponseKey`	Sollte bereitgestellt werden, um die bidirektionale Kommunikation für eine WebSocket Route zu initiieren. Zurzeit ist dieser Wert ausschließlich auf `$default` eingeschränkt.	
`RouteResponse.ModelSelectionExpression`	Schlüssel für `RouteResponse.RequestModels`	Wird derzeit nicht unterstützt.	

Entwicklung einer WebSocket API in API Gateway

Dieser Abschnitt enthält Details zu den API Gateway-Funktionen, die Sie bei der Entwicklung Ihrer API Gateway-APIs benötigen.

Während Sie Ihre API Gateway-API entwickeln, entscheiden Sie sich für eine Reihe von Merkmalen Ihrer API. Diese Eigenschaften hängen davon ab, wofür Ihre API verwendet werden soll. So könnte es beispielsweise sein, dass Sie es nur bestimmten Clients gestatten möchten, die API aufzurufen. Vielleicht soll die API aber auch für alle verfügbar sein. Vielleicht benötigen Sie einen API-Aufruf, um eine Lambda-Funktion auszuführen, eine Datenbankabfrage durchzuführen oder eine Anwendung aufzurufen.

Themen

Erstellen Sie eine WebSocket API in API Gateway

Sie können eine WebSocket API in der API Gateway Gateway-Konsole erstellen, indem Sie den Befehl AWS CLI create-api oder den `CreateApi` Befehl in einem AWS SDK verwenden. Die folgenden Verfahren zeigen, wie Sie eine neue WebSocket API erstellen.

> **Note**
>
> WebSocket APIs unterstützen nur TLS 1.2. Frühere TLS-Versionen werden nicht unterstützt.

Erstellen Sie eine WebSocket API mithilfe von AWS CLI Befehlen

Um eine WebSocket API mit dem zu erstellen, AWS CLI muss der Befehl create-api aufgerufen werden, wie im folgenden Beispiel gezeigt, der eine API mit dem `$request.body.action` Routenauswahlausdruck erstellt:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
 WEBSOCKET --route-selection-expression '$request.body.action'
```

Beispielausgabe:

```
{
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "Name": "myWebSocketApi3",
    "CreatedDate": "2018-11-15T06:23:51Z",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "'$request.body.action'",
    "ApiId": "aabbccddee"
}
```

Erstellen Sie eine WebSocket API mit der API Gateway Gateway-Konsole

Sie können eine WebSocket API in der Konsole erstellen, indem Sie das WebSocket Protokoll auswählen und der API einen Namen geben.

> **Important**
>
> Sobald Sie die API erstellt haben, können Sie das für sie ausgewählte Protokoll nicht mehr ändern. Es gibt keine Möglichkeit, eine WebSocket API in eine REST-API umzuwandeln oder umgekehrt.

So erstellen Sie eine WebSocket API mit der API Gateway Gateway-Konsole

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie Create API (API erstellen).
2. Wählen Sie unter WebSocket API die Option Build aus. Es werden nur regionale Endpunkte unterstützt.
3. Geben Sie unter API-Name den Namen Ihrer API ein.

4. Geben Sie unter Ausdruck für die Routenauswahl einen Wert ein. z. B. `$request.body.action`.

   Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter the section called “”.

5. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie Leere API erstellen aus, wenn Sie eine API ohne Routen erstellen möchten.
   - Klicken Sie auf Weiter, um Ihrer API Routen anzuhängen.

   Sie können Routen anhängen, nachdem Sie Ihre API erstellt haben.

Mit Routen für WebSocket APIs arbeiten

In Ihrer WebSocket API werden eingehende JSON-Nachrichten auf der Grundlage von Routen, die Sie konfigurieren, an Backend-Integrationen weitergeleitet. (Nicht-JSON-Nachrichten werden an eine von Ihnen konfigurierte `$default`-Route weitergeleitet.)

Eine Route umfasst einen Routenschlüssel. Dabei handelt es sich um den Wert, der bei der Auswertung eines Routen-Auswahlausdrucks erwartet wird. Das `routeSelectionExpression` ist ein auf API-Ebene definiertes Attribut. Es gibt eine JSON-Eigenschaft an, von der erwartet wird, dass sie in der Nachricht-Nutzlast vorhanden ist. Weitere Informationen zu Routen-Auswahlausdrücken finden Sie unter the section called “”.

Beispiel: Wenn Ihre JSON-Nachrichten eine `action`-Eigenschaft enthalten und Sie basierend auf dieser Eigenschaft verschiedene Aktionen durchführen möchten, könnte Ihr Routen-Auswahlausdruck `${request.body.action}` lauten. Ihre Routing-Tabelle würde in diesem Fall angeben, welche Aktion ausgeführt werden soll, indem der Wert der `action`-Eigenschaft gegen die benutzerdefinierten Routenschlüssel-Werte abgeglichen wird, die Sie in der Tabelle definiert haben.

Es gibt drei vordefinierte Routen, die verwendet werden können: `$connect`, `$disconnect` und `$default`. Darüber hinaus können Sie benutzerdefinierte Routen erstellen.

- API Gateway ruft die `$connect` Route auf, wenn eine persistente Verbindung zwischen dem Client und einer WebSocket API initiiert wird.
- API Gateway ruft die `$disconnect`-Route auf, wenn der Client oder der Server die Verbindung mit der API unterbricht.
- API Gateway ruft eine benutzerdefinierte Route auf, wenn nach der Auswertung des Routen-Auswahlausdrucks im Hinblick auf die Nachricht eine übereinstimmende Route gefunden wird. Die Übereinstimmung bestimmt, welche Integration aufgerufen wird.

- API Gateway ruft die Route `$default` auf, wenn der Routen-Auswahlausdruck nicht im Hinblick auf die Nachricht ausgewertet werden kann oder wenn keine übereinstimmende Route gefunden wird.

Routen-Auswahlausdrücke

Ein Routen-Auswahlausdruck wird ausgewertet, wenn der Service die Route auswählt, der eine eingehende Nachricht folgen soll. Der Service verwendet die Route, deren `routeKey` genau dem ausgewerteten Wert entspricht. Wenn bei fehlender Übereinstimmung eine Route mit dem Routenschlüssel `$default` vorhanden ist, wird diese Route ausgewählt. Wenn keine Routen dem ausgewählten Wert entsprechen und keine `$default`-Route vorhanden ist, gibt der Service einen Fehler zurück. Bei WebSocket basierten APIs sollte der Ausdruck die folgende Form haben`$request.body.`*`{path_to_body_element}`*.

Angenommen, Sie senden die folgende JSON-Nachricht:

```
{
    "service" : "chat",
    "action" : "join",
    "data" : {
        "room" : "room1234"
   }
}
```

Möglicherweise möchten Sie das Verhalten Ihrer API basierend auf der Eigenschaft `action` auswählen. In diesem Fall könnten Sie den folgenden Routen-Auswahlausdruck definieren:

```
$request.body.action
```

In diesem Beispiel bezieht sich `request.body` auf die JSON-Nutzlast Ihrer Nachricht und `.action` ist ein JSONPath-Ausdruck. Sie können nach `request.body` einen beliebigen JSON-Pfad-Ausdruck verwenden. Denken Sie aber daran, dass das Ergebnis in Text umgewandelt wird. Beispiel: Wenn Ihr JSONPath-Ausdruck ein Array mit zwei Elementen zurückgibt, werden sie als Zeichenfolge dargestell `"[item1, item2]"`. Aus diesem Grund ist es sinnvoll, dass Ihr Ausdruck auf einen Wert und nicht auf ein Array oder Objekt ausgewertet wird.

Sie können einfach einen statischen Wert oder auch mehrere Variablen verwenden. In der folgenden Tabelle werden Beispiele und deren im Hinblick auf die vorhergehende Nutzlast ausgewerteten Ergebnisse gezeigt.

Ausdruck	Ausgewertetes Ergebnis	Beschreibung
`$request.body.action`	`join`	Eine entpackte Variable
`${request.body.action}`	`join`	Eine verpackte Variable
`${request.body.service}/${request.body.action}`	`chat/join`	Mehrere Variablen mit statischen Werten
`${request.body.action}-${request.body.invalidPath}`	`join-`	Wenn der JSONPath nicht gefunden wird, wird die Variable als "" aufgelöst.
`action`	`action`	Statischer Wert
`\$default`	`$default`	Statischer Wert

Das ausgewertete Ergebnis wird zum Auffinden der Route verwendet. Wenn eine Route mit einem übereinstimmenden Routenschlüssel vorhanden ist, wird die Route zur Verarbeitung der Nachricht ausgewählt. Wenn keine passende Route gefunden wird, versucht API Gateway, die `$default`-Route zu finden, falls verfügbar. Wenn die `$default`-Route nicht definiert ist, gibt API Gateway einen Fehler zurück.

Routen für eine WebSocket API in API Gateway einrichten

Wenn Sie zum ersten Mal eine neue WebSocket API erstellen, gibt es drei vordefinierte Routen: `$connect``$disconnect`, und`$default`. Sie können sie mithilfe der Konsole, der API oder erstellen AWS CLI. Auf Wunsch können Sie benutzerdefinierte Routen erstellen. Weitere Informationen finden Sie unter the section called “Über WebSocket APIs”.

> **Note**
>
> In der CLI ist es möglich, Routen vor oder nach dem Erstellen von Integrationen zu erstellen. Auch können Sie dieselbe Integration für mehrere Routen wiederverwenden.

Route mit der API Gateway-Konsole erstellen

So erstellen Sie eine Route über die API Gateway-Konsole:

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie bei Route key (Routenschlüssel) den Schlüsselnamen ein. Sie können die vordefinierten Routen (`$connect`, `$disconnect` und `$default`) oder eine benutzerdefinierte Route erstellen.

   > **Note**
   >
   > Wenn Sie eine benutzerdefinierte Route erstellen, verwenden Sie nicht das $-Präfix im Routenschlüsselnamen. Dieses Präfix ist für vordefinierte Routen reserviert.

4. Wählen und konfigurieren Sie den Integrationstyp für die Route. Weitere Informationen finden Sie unter the section called “Richten Sie mit der WebSocket API Gateway-Konsole eine API-Integrationsanfrage ein”.

Erstellen Sie eine Route mit dem AWS CLI

Um eine Route mit dem zu erstellen AWS CLI, rufen Sie `create-route`wie im folgenden Beispiel gezeigt auf:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key
 $default
```

Beispielausgabe:

```
{
    "ApiKeyRequired": false,
    "AuthorizationType": "NONE",
    "RouteKey": "$default",
    "RouteId": "1122334"
}
```

Angeben der Routenanforderungs-Einstellungen für **`$connect`**

Wenn Sie die `$connect`-Route für Ihre API einrichten, sind die folgenden optionalen Einstellungen verfügbar, um die Autorisierung für Ihre API zu ermöglichen. Weitere Informationen finden Sie unter the section called “Die Route `$connect`”.

- Authorization (Autorisierung): Wenn keine Autorisierung erforderlich ist, können Sie `NONE` angeben. Geben Sie andernfalls Folgendes ein:
  - `AWS_IAM`um den Zugriff auf Ihre API mithilfe von AWS Standard-IAM-Richtlinien zu kontrollieren.
  - `CUSTOM`, um die Autorisierung für eine API durch Angabe einer zuvor von Ihnen erstellten Lambda-Genehmigerfunktion zu implementieren. Der Autorisierer kann sich in Ihrem eigenen AWS Konto oder einem anderen Konto befinden. AWS Weitere Informationen über Lambda-Genehmiger finden Sie unter API Gateway-Lambda-Genehmiger verwenden.

    > **Note**
    >
    > In der API Gateway-Konsole ist die Einstellung `CUSTOM` erst sichtbar, nachdem Sie eine Genehmigerfunktion wie unter the section called “Konfigurieren Sie einen Lambda-Autorisierer (Konsole)” beschrieben eingerichtet haben.

> **Important**
>
> Die Einstellung von Authorization (Autorisierung) wird auf die gesamte API angewendet, nicht nur auf die `$connect`-Route. Die `$connect`-Route schützt die übrigen Routen, da sie für jede Verbindung aufgerufen wird.

- API Key Required (API-Schlüssel erforderlich): Sie können für die `$connect`-Route einer API optional einen API-Schlüssel verlangen. Sie können API-Schlüssel zusammen mit Nutzungsplänen zur Kontrolle und Nachverfolgung des Zugriffs auf Ihre APIs einsetzen. Weitere Informationen finden Sie unter the section called “Nutzungspläne”.

`$connect`-Routenanforderung über die API Gateway-Konsole einrichten

So richten Sie die `$connect` Routenanforderung für eine WebSocket API mithilfe der API Gateway Gateway-Konsole ein:

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie unter Routes (Routen) `$connect` aus oder erstellen Sie eine `$connect`-Route, indem Sie wie folgt vorgehen the section called “Route mit der API Gateway-Konsole erstellen”.
3. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Authorization (Autorisierung) einen Autorisierungstyp aus.
5. Um eine API für die `$connect`-Route anzufordern, wählen Sie Require API key (API-Schlüssel erforderlich) aus.
6. Wählen Sie Änderungen speichern aus.

Richten Sie Routenantworten für eine WebSocket API in API Gateway ein

WebSocket Routen können für bidirektionale oder unidirektionale Kommunikation konfiguriert werden. API-Gateway übergibt die Backend-Antwort nicht über die Routenantwort, es sei denn, Sie haben eine Routenantwort eingerichtet.

> **Note**
>
> Sie können die `$default` Routenantwort nur für WebSocket APIs definieren. Sie können eine Integrationsantwort verwenden, um die Antwort von einem Backend-Service zu manipulieren. Weitere Informationen finden Sie unter the section called “Übersicht über Integrationsantworten”.

Sie können Routenantworten und Antwortauswahlausdrücke mithilfe der API Gateway Gateway-Konsole AWS CLI oder eines AWS SDK konfigurieren.

Weitere Informationen über Routenantwort-Auswahlausdrücke finden Sie unter the section called “”.

Themen

- Routenantwort mit der API Gateway-Konsole einrichten
- Richten Sie eine Routenantwort mit dem ein AWS CLI

Routenantwort mit der API Gateway-Konsole einrichten

Nachdem Sie eine WebSocket API erstellt und eine Proxy-Lambda-Funktion an die Standardroute angehängt haben, können Sie die Routenantwort mit der API Gateway Gateway-Konsole einrichten:

1. Melden Sie sich bei der API Gateway Gateway-Konsole an und wählen Sie eine WebSocket API mit einer Proxy-Lambda-Funktionsintegration auf der `$default` Route aus.
2. Wählen Sie unter Routes (Routen) die `$default`-Route aus.
3. Wählen Sie Enable two-way communication (Bidirektionale Kommunikation aktivieren).
4. Klicken Sie auf Deploy API.
5. Stellen Sie Ihre API für eine Stufe bereit.

Stellen Sie mit dem folgenden wscat-Befehl eine Verbindung mit Ihrer API her. Mehr über `wscat` erfahren Sie unter the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Drücken Sie die Eingabetaste, um die Standardroute aufzurufen. Der Hauptteil Ihrer Lambda-Funktion sollte zurückgegeben werden.

Richten Sie eine Routenantwort mit dem ein AWS CLI

Um eine Routenantwort für eine WebSocket API mithilfe von einzurichten AWS CLI, rufen Sie den `create-route-response`Befehl auf, wie im folgenden Beispiel gezeigt. Sie können die API-ID und Routing-ID identifizieren, indem Sie `get-apis` und `get-routes` aufrufen.

```
aws apigatewayv2 create-route-response \
    --api-id aabbccddee \
    --route-id 1122334  \
    --route-response-key '$default'
```

Beispielausgabe:

```
{
    "RouteResponseId": "abcdef",
    "RouteResponseKey": "$default"
}
```

Einrichtung einer **`$connect`** Route, für die ein WebSocket Unterprotokoll erforderlich ist

Kunden können das `Sec-WebSocket-Protocol` Feld verwenden, um während der Verbindung zu Ihrer WebSocket API ein WebSocket Unterprotokoll anzufordern. Sie können eine Integration für die `$connect`-Route einrichten, um Verbindungen nur zuzulassen, wenn ein Client ein von Ihrer API unterstütztes Unterprotokoll anfordert.

Die folgende Lambda-Beispielfunktion gibt den `Sec-WebSocket-Protocol`-Header an Clients zurück. Die Funktion stellt nur dann eine Verbindung zu Ihrer API her, wenn der Client das `myprotocol`-Unterprotokoll angibt.

Eine AWS CloudFormation Vorlage, die diese Beispiel-API- und Lambda-Proxyintegration erstellt, finden Sie unter `ws-subprotocol.yaml`.

```
export const handler = async (event) => {
    if (event.headers != undefined) {
        const headers = toLowerCaseProperties(event.headers);

        if (headers['sec-websocket-protocol'] != undefined) {
            const subprotocolHeader = headers['sec-websocket-protocol'];
            const subprotocols = subprotocolHeader.split(',');
```

```
            if (subprotocols.indexOf('myprotocol') >= 0) {
                const response = {
                    statusCode: 200,
                    headers: {
                        "Sec-WebSocket-Protocol" : "myprotocol"
                    }
                };
                return response;
            }
        }
    }

    const response = {
        statusCode: 400
    };

    return response;
};

function toLowerCaseProperties(obj) {
    var wrapper = {};
    for (var key in obj) {
        wrapper[key.toLowerCase()] = obj[key];
    }
    return wrapper;
}
```

Sie können [wscat](#) verwenden, um zu testen, ob Ihre API Verbindungen nur zulässt, wenn ein Client ein von Ihrer API unterstütztes Unterprotokoll anfordert. Die folgenden Befehle verwenden das `-s`-Flag, um während der Verbindung Unterprotokolle anzugeben.

Mit dem folgenden Befehl wird versucht, eine Verbindung mit einem nicht unterstützten Unterprotokoll herzustellen. Da der Client das `chat1`-Unterprotokoll angegeben hat, gibt die Lambda-Integration einen 400-Fehler zurück und die Verbindung ist nicht erfolgreich.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

Der folgende Befehl enthält ein unterstütztes Unterprotokoll in der Verbindungsanforderung. Die Lambda-Integration ermöglicht die Verbindung.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Weitere Informationen zum Aufrufen von WebSocket APIs finden Sie unter. Eine WebSocket API aufrufen

Steuern und Verwalten des Zugriffs auf eine WebSocket API in API Gateway

API Gateway unterstützt mehrere Mechanismen zur Steuerung und Verwaltung des Zugriffs auf Ihre WebSocket API.

Sie können die folgenden Mechanismen für die Authentifizierung und Autorisierung verwenden:

- AWS Standard-IAM-Rollen und -Richtlinien bieten flexible und robuste Zugriffskontrollen. Sie können IAM-Rollen und -Richtlinien verwenden, um zu steuern, wer Ihre APIs erstellen und verwalten kann und wer sie aufrufen kann. Weitere Informationen finden Sie unter IAM-Genehmiger verwenden.
- IAM-Tags können zusammen mit IAM-Richtlinien verwendet werden, um den Zugriff zu steuern. Weitere Informationen finden Sie unter Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden.
- Lambda-Genehmiger sind Lambda-Funktionen, die den Zugriff auf APIs steuern. Weitere Informationen finden Sie unter Lambda REQUEST-Genehmigerfunktion erstellen.

Themen

IAM-Genehmiger verwenden

Die IAM-Autorisierung in WebSocket APIs ähnelt der für REST-APIs, mit den folgenden Ausnahmen:

- Die Aktion `execute-api` unterstützt `ManageConnections` zusätzlich zu vorhandenen Aktionen (`Invoke`, `InvalidateCache`). `ManageConnections` steuert den Zugriff auf die API @connections.
- WebSocket Routen verwenden ein anderes ARN-Format:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- Die API `@connections` verwendet das gleiche ARN-Format wie REST-APIs:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

> **Important**
>
> Wenn Sie die IAM-Autorisierung verwenden, müssen Sie Anforderungen mit Signature Version 4 (SigV4) signieren.

Sie könnten z. B. die folgende Richtlinie für den Client einrichten. Dieses Beispiel ermöglicht allen Benutzern das Senden einer Nachricht (`Invoke`) für alle Routen, außer für eine geheime Route in der Stufe `prod`, und hindert alle Benutzer am Senden einer Nachricht zurück an verbundene Clients (`ManageConnections`) für alle Stufen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/prod/*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/prod/secret"
      ]
    },
    {
```

```
      "Effect": "Deny",
      "Action": [
        "execute-api:ManageConnections"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*"
      ]
    }
  ]
}
```

Lambda **REQUEST**-Genehmigerfunktion erstellen

Eine Lambda-Autorisierungsfunktion in WebSocket APIs ähnelt der für REST-APIs, mit den folgenden Ausnahmen:

- Sie können für die `$connect`-Route nur eine Lambda-Genehmiger-Funktion verwenden.
- Sie können keine Pfadvariablen (`event.pathParameters`) verwenden, da der Pfad fest ist.
- `event.methodArn` unterscheidet sich von seinem REST API-Äquivalent, da es über keine HTTP-Methode verfügt. Im Fall von `$connect` endet `methodArn` mit `"$connect"`:

  ```
  arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
  ```

- Die Kontextvariablen in `event.requestContext` unterscheiden sich von denen für REST-APIs.

Das folgende Beispiel zeigt eine Eingabe für einen `REQUEST` Autorisierer für eine API: WebSocket

```
{
    "type": "REQUEST",
    "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/default/
$connect",
    "headers": {
        "Connection": "upgrade",
        "content-length": "0",
        "HeaderAuth1": "headerValue1",
        "Host": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "Sec-WebSocket-Extensions": "permessage-deflate; client_max_window_bits",
        "Sec-WebSocket-Key": "...",
        "Sec-WebSocket-Version": "13",
        "Upgrade": "websocket",
        "X-Amzn-Trace-Id": "...",
```

```
        "X-Forwarded-For": "...",
        "X-Forwarded-Port": "443",
        "X-Forwarded-Proto": "https"
    },
    "multiValueHeaders": {
        "Connection": [
            "upgrade"
        ],
        "content-length": [
            "0"
        ],
        "HeaderAuth1": [
            "headerValue1"
        ],
        "Host": [
            "abcdef123.execute-api.us-east-1.amazonaws.com"
        ],
        "Sec-WebSocket-Extensions": [
            "permessage-deflate; client_max_window_bits"
        ],
        "Sec-WebSocket-Key": [
            "..."
        ],
        "Sec-WebSocket-Version": [
            "13"
        ],
        "Upgrade": [
            "websocket"
        ],
        "X-Amzn-Trace-Id": [
            "..."
        ],
        "X-Forwarded-For": [
            "..."
        ],
        "X-Forwarded-Port": [
            "443"
        ],
        "X-Forwarded-Proto": [
            "https"
        ]
    },
    "queryStringParameters": {
        "QueryString1": "queryValue1"
```

```
    },
    "multiValueQueryStringParameters": {
        "QueryString1": [
            "queryValue1"
        ]
    },
    "stageVariables": {},
    "requestContext": {
        "routeKey": "$connect",
        "eventType": "CONNECT",
        "extendedRequestId": "...",
        "requestTime": "19/Jan/2023:21:13:26 +0000",
        "messageDirection": "IN",
        "stage": "default",
        "connectedAt": 1674162806344,
        "requestTimeEpoch": 1674162806345,
        "identity": {
            "sourceIp": "..."
        },
        "requestId": "...",
        "domainName": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "connectionId": "...",
        "apiId": "abcdef123"
    }
}
```

Die folgende Lambda-Autorisierungsfunktion ist eine WebSocket Version der Lambda-Autorisierungsfunktion für REST-APIs in: the section called “Zusätzliche Beispiele für Lambda-Autorisierungsfunktionen”

Node.js

```
  // A simple REQUEST authorizer example to demonstrate how to use request
  // parameters to allow or deny a request. In this example, a request is
  // authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
  // in the request context match the specified values of
  // of 'headerValue1' and 'queryValue1' respectively.
           export const handler = function(event, context, callback) {
   console.log('Received event:', JSON.stringify(event, null, 2));

  // Retrieve request parameters from the Lambda function input:
  var headers = event.headers;
```

```
    var queryStringParameters = event.queryStringParameters;
    var stageVariables = event.stageVariables;
    var requestContext = event.requestContext;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var ApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var route = apiGatewayArnTmp[2];

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
     condition.IpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1") {
         callback(null, generateAllow('me', event.methodArn));
     }  else {
         callback("Unauthorized");
     }
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
     authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
         policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
         statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
         statementOne.Resource = resource;
         policyDocument.Statement[0] = statementOne;
         authResponse.policyDocument = policyDocument;
     }
    // Optional output with custom properties of the String, Number or Boolean type.
```

```
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
     };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
 parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    ApiId = apiGatewayArnTmp[0]
```

```
    stage = apiGatewayArnTmp[1]
    route = apiGatewayArnTmp[2]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}

    if (headers['HeaderAuth1'] ==
            "headerValue1" and queryStringParameters["QueryString1"] ==
 "queryValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
        print('unauthorized')
        return 'unauthorized'

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)
```

```
    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

Gehen Sie genauso vor wie bei REST-APIs, um die vorherige Lambda-Funktion als `REQUEST` Autorisierungsfunktion für eine WebSocket API zu konfigurieren.

Um die `$connect`-Route für die Verwendung dieses Lambda-Genehmigers in der Konsole zu konfigurieren, wählen Sie die `$connect`-Route oder erstellen diese. Wählen Sie im Abschnitt Route request settings (Einstellungen der Routenanforderung) die Option Edit (Bearbeiten) aus. Wählen Sie im Dropdown-Menü Authorization (Autorisierung) Ihren Autorisierer aus und wählen Sie dann Save changes (Änderungen speichern) aus.

Zum Testen des Genehmigers müssen Sie eine neue Verbindung erstellen. Wenn der Genehmiger in `$connect` geändert wird, hat dies keine Auswirkungen auf den bereits verbundenen Client. Wenn Sie eine Verbindung zu Ihrer WebSocket API herstellen, müssen Sie Werte für alle konfigurierten Identitätsquellen angeben. Beispielsweise können Sie eine Verbindung durch Senden einer gültigen Abfragezeichenfolge und eines gültigen Headers unter Verwendung von `wscat` wie im folgenden Beispiel herstellen:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Bei dem Versuch, ohne einen gültigen Identitätswert eine Verbindung herzustellen, erhalten Sie eine `401`-Antwort:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

WebSocket API-Integrationen einrichten

Nach dem Einrichten einer API-Route müssen Sie diese in einen Endpunkt im Backend integrieren. Ein Backend-Endpunkt wird auch als Integrationsendpunkt bezeichnet und kann eine Lambda-Funktion, ein HTTP-Endpunkt oder eine AWS Serviceaktion sein. Die API-Integration beinhaltet eine Integrationsanforderung und eine Integrationsantwort.

In diesem Abschnitt erfahren Sie, wie Sie Integrationsanfragen und Integrationsantworten für Ihre WebSocket API einrichten.

Themen

Einrichtung einer WebSocket API-Integrationsanfrage in API Gateway

Das Einrichten einer Integrationsanforderung umfasst die folgenden Schritte:

- Auswählen eines Routenschlüssels zum Integrieren im Backend.
- Angabe des aufzurufenden Backend-Endpunkts. WebSocket APIs unterstützen die folgenden Integrationstypen:
  - `AWS_PROXY`
  - `AWS`
  - `HTTP_PROXY`
  - `HTTP`
  - `MOCK`

  Weitere Informationen zu Integrationstypen finden Sie IntegrationTypein der API Gateway V2 REST API.
- Konfigurieren, wie die Routenanforderungsdaten bei Bedarf in Integrationsanforderungsdaten konvertiert werden, durch Angabe einer oder mehrerer Anforderungsvorlagen.

Richten Sie mit der WebSocket API Gateway-Konsole eine API-Integrationsanfrage ein

So fügen Sie mithilfe der API-Gateway-Konsole eine Integrationsanfrage zu einer Route in einer WebSocket API hinzu

1. Melden Sie sich bei der API Gateway-Konsole an, wählen Sie die API und wählen Sie Routes (Routen).
2. Wählen Sie unter Routes (Routen) die Route aus.
3. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Edit (Bearbeiten) aus.
4. Wählen Sie für Integration type (Integrationstyp) eine der folgenden Optionen:

   - Wählen Sie die Lambda-Funktion nur, wenn Ihre API in eine AWS Lambda Funktion integriert wird, die Sie bereits in diesem Konto oder in einem anderen Konto erstellt haben.

     Um eine neue Lambda-Funktion in zu erstellen AWS Lambda, eine Ressourcenberechtigung für die Lambda-Funktion festzulegen oder andere Lambda-Serviceaktionen auszuführen, wählen Sie AWS stattdessen Service.
   - Wählen Sie HTTP, wenn Ihre API in einen vorhandenen HTTP-Endpunkt integriert wird. Weitere Informationen finden Sie unter HTTP-Integrationen in API Gateway einrichten.
   - Wählen Sie Mock, wenn Sie API-Antworten direkt von API Gateway generieren möchten, ohne dass ein Integrations-Backend erforderlich ist. Weitere Informationen finden Sie unter Mock-Integrationen in API Gateway einrichten.
   - Wählen Sie AWS Service, wenn Ihre API in einen Dienst integriert werden soll. AWS
   - Wählen Sie VPC Link (VPC-Link), wenn Ihre API einen `VpcLink` als privaten Integrations-Endpunkt verwenden wird. Weitere Informationen finden Sie unter Private API Gateway-Integrationen einrichten.
5. Wenn Sie Lambda Function (Lambda-Funktion) wählen, gehen Sie wie folgt vor:

   a. Markieren Sie unter Use Lambda Proxy integration (Lambda-Proxy-Integration verwenden) das Kontrollkästchen, wenn Sie die Lambda-Proxy-Integration oder die kontoübergreifende Lambda-Proxy-Integration verwenden möchten.

   b. Geben Sie für Lambda Function (Lambda-Funktion) die Funktion auf eine der folgenden Arten an:

- Wenn sich Ihre Lambda-Funktion in demselben Konto befindet, geben Sie den Funktionsnamen ein und wählen Sie dann die Funktion aus der Dropdown-Liste aus.

  > **Note**
  >
  > Der Funktionsname kann optional seinen Alias oder seine Versionsangabe, wie in `HelloWorld`, `HelloWorld:1` oder `HelloWorld:alpha`, enthalten.

- Wenn sich die Funktion in einem anderen Konto befindet, geben Sie den ARN für die Funktion ein.

c. Wenn Sie den Timeout-Standardwert von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen `50` und `29000` Millisekunden ein.

6. Befolgen Sie bei Wahl von HTTP die Anweisungen in Schritt 4 von the section called “ Einrichten einer API-Integrationsanforderung mit der Konsole”.
7. Befolgen Sie bei Wahl von Mock (Pseudo) die Anweisungen unter dem Schritt Request Templates (Vorlagen anfordern).
8. Befolgen Sie bei Wahl von AWS -Service die Anweisungen in Schritt 6 von the section called “ Einrichten einer API-Integrationsanforderung mit der Konsole”.
9. Wenn Sie VPC Link (VPC-Link) ausgewählt haben, gehen Sie wie folgt vor:

   a. Markieren Sie für VPC proxy integration (Proxy-Integration verwenden) das Kontrollkästchen, wenn Anforderungen über einen Proxy an Ihren `VPCLink`-Endpunkt gesendet werden sollen.

   b. Wählen Sie als HTTP method die HTTP-Methode aus, die am ehesten der Methode im HTTP-Backend entspricht.

   c. Wählen Sie in der Dropdownliste VPC link (VPC-Link) einen VPC-Link aus. Sie können `[Use Stage Variables]` auswählen und **`${stageVariables.vpcLinkId}`** in das Textfeld unter der Liste eingeben.

      Sie können die `vpcLinkId`-Stufenvariable nach der Bereitstellung der API für eine Stufe definieren und ihren Wert auf die ID des `VpcLink` festlegen.

d. Geben Sie als Endpunkt-URL die URL des HTTP-Backends ein, das von dieser Integration verwendet werden soll.

e. Wenn Sie die Standardzeitüberschreitung von 29 Sekunden verwenden möchten, lassen Sie das Kontrollkästchen Default timeout (Standardzeitüberschreitung) aktiviert. Wenn Sie einen benutzerdefinierten Zeitüberschreitungswert festlegen möchten, wählen Sie Default timeout (Standardzeitüberschreitung) aus und geben Sie einen Zeitüberschreitungswert zwischen `50` und `29000` Millisekunden ein.

10. Wählen Sie Änderungen speichern aus.

11. Führen Sie unter Request Templates (Vorlagen anfordern) die folgenden Schritte durch:

    a. Um einen Template selection expression (Vorlagen-Auswahlausdruck) einzugeben, wählen Sie unter Request templates (Vorlagen anfordern) die Option Edit (Bearbeiten).

    b. Geben Sie einen Template selection expression (Vorlagen-Auswahlausdruck) ein. Verwenden Sie einen Ausdruck, nach dem API Gateway in der Nachrichtennutzlast sucht. Wenn er gefunden wird, wird er ausgewertet. Das Ergebnis ist ein Vorlagen-Schlüsselwert zur Auswahl der Datenzuweisungsvorlage, die auf die Daten in der Nachrichtennutzlast anzuwenden ist. Die Datenzuweisungsvorlage erstellen Sie im nächsten Schritt. Wählen Sie Edit (Bearbeiten) aus, um Ihre Änderungen zu speichern.

    c. Wählen Sie Create template (Vorlage erstellen), um die Datenzuweisungsvorlage zu erstellen. Geben Sie für Template key (Vorlagenschlüssel) einen Vorlagenschlüsselwert zur Auswahl der Datenzuweisungsvorlage ein, die auf die Daten in der Nachrichtennutzlast anzuwenden ist. Definieren Sie anschließend eine Zuweisungsvorlage. Wählen Sie Create template (Vorlage erstellen) aus.

       Weitere Informationen zu Vorlagen-Auswahlausdrücken finden Sie unter the section called “Vorlagen-Auswahlausdrücke”.

Richten Sie eine Integrationsanfrage ein, indem Sie AWS CLI

Sie können eine Integrationsanfrage für eine Route in einer WebSocket API einrichten, indem Sie AWS CLI wie im folgenden Beispiel vorgehen, wodurch eine Scheinintegration erstellt wird:

1. Erstellen Sie eine Datei mit dem Namen `integration-params.json` und dem folgenden Inhalt:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Führen Sie den Befehl create-integration wie im folgenden Beispiel gezeigt aus:

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccddee --cli-
input-json file://integration-params.json
```

Dies ist die Beispielsausgabe dieses Befehls:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationResponseSelectionExpression": "${response.statuscode}",
    "RequestTemplates": {
        "application/json": "{\"statusCode\":200}"
    },
    "IntegrationId": "0abcdef",
    "IntegrationType": "MOCK"
}
```

Alternativ können Sie eine Integrationsanforderung für eine Proxyintegration einrichten, indem Sie den Befehl AWS CLI wie im folgenden Beispiel verwenden:

1. Erstellen Sie eine Lambda-Funktion in der Lambda-Konsole und geben Sie ihr eine grundlegende Lambda-Ausführungsrolle.
2. Führen Sie den Befehl create-integration wie im folgenden Beispiel aus:

```
aws apigatewayv2 create-integration --api-id aabbccddee --integration-type
 AWS_PROXY --integration-method POST --integration-uri arn:aws:apigateway:us-
east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
east-1:123412341234:function:simpleproxy-echo-e2e/invocations
```

Dies ist die Beispielsausgabe dieses Befehls:

```
{
```

```
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "IntegrationMethod": "POST",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationUri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:simpleproxy-echo-e2e/invocations",
    "IntegrationId": "abcdefg",
    "IntegrationType": "AWS_PROXY"
}
```

Eingabeformat einer Lambda-Funktion für die Proxyintegration für APIs WebSocket

Bei der Lambda-Proxy-Integration ordnet API Gateway die gesamte Client-Anforderung dem `event`-Eingabeparameter der Backend-Lambda-Funktion zu. Das folgende Beispiel zeigt die Struktur des Eingabeereignisses von der `$connect` Route und des Eingabeereignisses von der `$disconnect` Route, das API Gateway an eine Lambda-Proxyintegration sendet.

Input from the $connect route

```
{
    headers: {
      Host: 'abcd123.execute-api.us-east-1.amazonaws.com',
      'Sec-WebSocket-Extensions': 'permessage-deflate; client_max_window_bits',
      'Sec-WebSocket-Key': '...',
      'Sec-WebSocket-Version': '13',
      'X-Amzn-Trace-Id': '...',
      'X-Forwarded-For': '192.0.2.1',
      'X-Forwarded-Port': '443',
      'X-Forwarded-Proto': 'https'
    },
    multiValueHeaders: {
      Host: [ 'abcd123.execute-api.us-east-1.amazonaws.com' ],
      'Sec-WebSocket-Extensions': [ 'permessage-deflate; client_max_window_bits' ],
      'Sec-WebSocket-Key': [ '...' ],
      'Sec-WebSocket-Version': [ '13' ],
      'X-Amzn-Trace-Id': [ '...' ],
      'X-Forwarded-For': [ '192.0.2.1' ],
      'X-Forwarded-Port': [ '443' ],
      'X-Forwarded-Proto': [ 'https' ]
    },
    requestContext: {
      routeKey: '$connect',
      eventType: 'CONNECT',
```

```
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:11:43 +0000',
      messageDirection: 'IN',
      stage: 'prod',
      connectedAt: 1707502303419,
      requestTimeEpoch: 1707502303420,
      identity: { sourceIp: '192.0.2.1' },
      requestId: 'ABCD1234=',
      domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      connectionId: 'AAAA1234=',
      apiId: 'abcd1234'
    },
    isBase64Encoded: false
  }
```

Input from the $disconnect route

```
{
    headers: {
      Host: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      'x-api-key': '',
      'X-Forwarded-For': '',
      'x-restapi': ''
    },
    multiValueHeaders: {
      Host: [ 'abcd1234.execute-api.us-east-1.amazonaws.com' ],
      'x-api-key': [ '' ],
      'X-Forwarded-For': [ '' ],
      'x-restapi': [ '' ]
    },
    requestContext: {
      routeKey: '$disconnect',
      disconnectStatusCode: 1005,
      eventType: 'DISCONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:23:28 +0000',
      messageDirection: 'IN',
      disconnectReason: 'Client-side close frame status not set',
      stage: 'prod',
      connectedAt: 1707503007396,
      requestTimeEpoch: 1707503008941,
      identity: { sourceIp: '192.0.2.1' },
```

```
    requestId: 'ABCD1234=',
    domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
    connectionId: 'AAAA1234=',
    apiId: 'abcd1234'
  },
  isBase64Encoded: false
}
```

WebSocket API-Integrationsantworten in API Gateway einrichten

Themen

- Übersicht über Integrationsantworten
- Integrationsantworten für bidirektionale Kommunikation
- Integrationsantwort mit der API Gateway-Konsole einrichten
- Richten Sie eine Integrationsantwort mit dem ein AWS CLI

Übersicht über Integrationsantworten

Die Integrationsantwort von API Gateway ist eine Möglichkeit, die Antwort von einem Backend-Service zu modellieren und zu manipulieren. Es gibt einige Unterschiede bei der Einrichtung einer REST-API im Vergleich zu einer WebSocket API-Integrationsantwort, aber das Verhalten ist konzeptionell dasselbe.

WebSocket Routen können für bidirektionale oder unidirektionale Kommunikation konfiguriert werden.

- Wenn für eine Route bidirektionale Kommunikation konfiguriert ist, ermöglicht Ihnen eine Integrationsantwort, Transformationen für die zurückgegebene Nachrichtennutzlast zu konfigurieren, ähnlich wie Integrationsantworten für REST-APIs.
- Wenn eine Route für die unidirektionale Kommunikation konfiguriert ist, wird unabhängig von der Konfiguration der Integrationsantwort nach der Verarbeitung der Nachricht keine Antwort über den WebSocket Kanal zurückgegeben.

API Gateway übergibt die Backend-Antwort nicht über die Routenantwort, es sei denn, Sie richten eine Routenantwort ein. Weitere Informationen zum Einrichten einer Routenreaktion finden Sie unter the section called “Richten Sie WebSocket API-Routenantworten ein”.

Integrationsantworten für bidirektionale Kommunikation

Integrationen lassen sich in Proxy-Integrationen und Nicht-Proxy-Integrationen unterteilen.

> **Important**
>
> Bei Proxy-Integrationen gibt API Gateway automatisch die Backend-Ausgabe als vollständigen Payload an den Aufrufer weiter. Es gibt keine Integrationsantwort.

Bei Nicht-Proxy-Integrationen müssen Sie mindestens eine Integrationsantwort einrichten:

- Im Idealfall sollte eine Ihrer Integrationsantworten als Catch-all-Methode dienen, wenn keine explizite Wahl vorgenommen werden kann. Dieser Standardfall wird durch Festlegen des Integrationsantwort-Schlüssels dargestell `$default`.
- In allen anderen Fällen fungiert der Integrationsantwort-Schlüssel als regulärer Ausdruck. Er sollte dem Format folge `"/expression/"`.

Bei Nicht-Proxy-HTTP-Integrationen:

- API Gateway versucht, den HTTP-Statuscode der Backend-Antwort zuzuordnen. Der Integrationsantwort-Schlüssel fungiert in diesem Fall als regulärer Ausdruck. Wenn keine Übereinstimmung gefunden wird, dann wird `$default` als Integrationsantwort gewählt.
- Der Vorlagen-Auswahlausdruck funktioniert wie oben beschrieben identisch. Zum Beispiel:
  - `/2\d\d/`: Erfolgreiche Antworten empfangen und transformieren
  - `/4\d\d/`: Fehler (ungültige Anforderung) empfangen und transformieren
  - `$default`: Alle unerwarteten Antworten empfangen und transformieren

Weitere Informationen zu Vorlagen-Auswahlausdrücken finden Sie unter the section called “Vorlagen-Auswahlausdrücke”.

Integrationsantwort mit der API Gateway-Konsole einrichten

So richten Sie mithilfe der API-Gateway-Konsole eine Antwort zur Routenintegration für eine WebSocket API ein:

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.

2. Wählen Sie Ihre WebSocket API und wählen Sie Ihre Route.
3. Wählen Sie die Registerkarte Integration request (Integrationsanforderung) und dann im Abschnitt Integration request settings (Einstellungen für Integrationsanforderungen) die Option Create integration response (Integrationsantwort erstellen) aus.
4. Geben Sie für Antwortschlüssel einen Wert ein, der nach Auswertung des Ausdrucks der Antwortauswahl im Antwortschlüssel der ausgehenden Nachricht zu finden ist. Sie können beispielsweise **`/4\d\d/`** eingeben, um Anforderungsfehler zu empfangen und umzuwandeln, oder Sie können **`$default`** eingeben, um alle Antworten zu empfangen und umzuwandeln, die dem Ausdruck für die Vorlagenauswahl entsprechen.
5. Geben Sie unter Template selection expression (Ausdruck für die Vorlagenauswahl) einen Auswahlausdruck ein, um die ausgehende Nachricht auszuwerten.
6. Wählen Sie Create response (Antwort erstellen) aus.
7. Sie können auch eine Zuordnungsvorlage definieren, um Transformationen Ihrer Payload für zurückgegebene Nachrichten zu konfigurieren. Wählen Sie Create template (Vorlage erstellen) aus.
8. Geben Sie einen Schlüsselnamen ein. Wenn Sie den Standardausdruck für die Vorlagenauswahl auswählen, geben Sie **`\$default`** ein.
9. Geben Sie unter Response template (Antwortvorlage) Ihre Zuordnungsvorlage in den Code-Editor ein.
10. Wählen Sie Create template (Vorlage erstellen) aus.
11. Wählen Sie Deploy API (API bereitstellen) aus, um Ihre API bereitzustellen.

Stellen Sie mit dem folgenden wscat-Befehl eine Verbindung mit Ihrer API her. Mehr über `wscat` erfahren Sie unter the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Wenn Sie Ihre Route aufrufen, sollte die Payload der zurückgegebenen Nachricht zurückgegeben werden.

Richten Sie eine Integrationsantwort mit dem ein AWS CLI

Um eine Integrationsantwort für eine WebSocket API einzurichten, AWS CLI rufen Sie den create-integration-responseBefehl auf. Der folgende CLI-Befehl zeigt ein Beispiel für das Erstellen einer `$default`-Integrationsantwort:

```
aws apigatewayv2 create-integration-response \
    --api-id vaz7da96z6 \
    --integration-id a1b2c3 \
    --integration-response-key '$default'
```

Anforderungsvalidierung

Sie können API Gateway so konfigurieren, dass die Validierung einer Routenanforderung durchgeführt wird, bevor Sie mit der Integrationsanforderung fortfahren. Wenn die Validierung fehlschlägt, schlägt API Gateway die Anfrage fehl, ohne Ihr Backend aufzurufen, sendet eine Gateway-Antwort „Bad Request Body“ an den Client und veröffentlicht die Validierungsergebnisse in CloudWatch Logs. Die Verwendung der Validierung auf diese Weise reduziert unnötige Aufrufe an Ihr API-Backend.

Modell-Auswahlausdrücke

Sie können einen Modell-Auswahlausdruck verwenden, um Anforderungen innerhalb derselben Route dynamisch zu validieren. Die Modellvalidierung erfolgt, wenn Sie einen Modell-Auswahlausdruck für Proxy- oder Nicht-Proxy-Integrationen bereitstellen. Möglicherweise müssen Sie das `$default`-Modell als Fallback definieren, wenn kein passendes Modell gefunden wird. Wenn kein übereinstimmendes Modell vorhanden ist und `$default` nicht definiert ist, schlägt die Validierung fehl. Der Auswahlausdruck ähnelt `Route.ModelSelectionExpression` und wertet den Schlüssel für `Route.RequestModels` aus.

Wenn Sie eine Route für eine WebSocket API definieren, können Sie optional einen Modellauswahlausdruck angeben. Dieser Ausdruck wird zur Auswahl des Modells ausgewertet, das bei Eingang einer Anforderung für die Textvalidierung verwendet werden soll. Der Ausdruck wird auf einen der Einträge in den einer Route ausgewertet `requestmodels`.

Ein Modell wird als JSON-Schema ausgedrückt und beschreibt die Datenstruktur des Anforderungstextes. Die Beschaffenheit dieses Auswahlausdrucks ermöglicht Ihnen, dynamisch das Modell auszuwählen, über das zur Laufzeit für eine bestimmte Route validiert werden soll.

Weitere Informationen, wie Sie ein Modell erstellen können, finden Sie unter the section called “Datenmodelle”.

Einrichten der grundlegenden Anforderungsvalidierung über die API-Gateway-Konsole

Das folgende Beispiel zeigt Ihnen, wie Sie die Anforderungsvalidierung für eine Route einrichten.

Zuerst erstellen Sie ein Modell und dann eine Route. Als Nächstes konfigurieren Sie die Anforderungsvalidierung für die Route, die Sie gerade erstellt haben. Schließlich stellen Sie Ihre API bereit und testen sie. Um dieses Tutorial abzuschließen, benötigen Sie eine WebSocket API `$request.body.action` als Routenauswahlausdruck und einen Integrationsendpunkt für Ihre neue Route.

Sie benötigen außerdem `wscat`, um eine Verbindung zu Ihrer API herzustellen. Weitere Informationen finden Sie unter the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.

So erstellen Sie ein Modell

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie eine WebSocket API.
3. Klicken Sie im Navigationsbereich auf Models (Modelle).
4. Wählen Sie Modell erstellen aus.
5. Geben Sie unter Name **`emailModel`** ein.
6. Geben Sie für Content type (Inhaltstyp) **`application/json`** ein.
7. Geben Sie für Modellschema Folgendes ein:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type" : "object",
    "required" : [ "address"],
    "properties" : {
        "address": {
            "type": "string"
        }
    }
}
```

Dieses Modell erfordert, dass die Anfrage eine E-Mail-Adresse enthält.

8. Wählen Sie Speichern.

In diesem Schritt erstellen Sie eine Route für Ihre WebSocket API.

Erstellen Sie eine Route.

1. Wählen Sie im Hauptnavigationsbereich Routes aus.
2. Wählen Sie Create route (Route erstellen) aus.
3. Geben Sie in Route key (Routenschlüssel) **`sendMessage`** ein.
4. Wählen Sie einen Integrationstyp und geben Sie einen Integrationsendpunkt an. Weitere Informationen finden Sie unter the section called “Integrationen”.
5. Wählen Sie Create route (Route erstellen) aus.

In diesem Schritt richten Sie die Anforderungsvalidierung für die `sendMessage` Route ein.

Um die Anforderungsvalidierung einzurichten

1. Wählen Sie auf der Registerkarte Routenanfrage unter Einstellungen für Routenanfragen die Option Bearbeiten aus.
2. Geben Sie als Ausdruck für die Modellauswahl den Wert ein**`${request.body.messageType}`**.

   API Gateway verwendet die `messageType` Eigenschaft, um die eingehende Anfrage zu validieren.
3. Wählen Sie Anforderungsmodell hinzufügen aus.
4. Geben Sie als Modellschlüssel ein**`email`**.
5. Wählen Sie als Modell die Option EmailModel aus.

   API Gateway validiert eingehende Nachrichten, bei denen die `messageType` Eigenschaft auf gesetzt ist, `email` anhand dieses Modells.

> **Note**
>
> Wenn API Gateway den Modellauswahlausdruck nicht mit einem Modellschlüssel abgleichen kann, wählt es das `$default` Modell aus. Wenn kein `$default` Modell

vorhanden ist, schlägt die Validierung fehl. Für Produktions-APIs empfehlen wir, dass Sie ein `$default` Modell erstellen.

6. Wählen Sie Änderungen speichern aus.

In diesem Schritt stellen Sie Ihre API bereit und testen sie.

Um Ihre API bereitzustellen und zu testen

1. Klicken Sie auf Deploy API.
2. Wählen Sie die gewünschte Stufe aus der Dropdown-Liste aus oder geben Sie den Namen einer neuen Stufe ein.
3. Wählen Sie Bereitstellen.
4. Klicken Sie im Hauptnavigationsbereich auf Stages (Stufen).
5. Kopieren Sie die WebSocket URL Ihrer API. Die URL sollte wie `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production` aussehen.
6. Öffnen Sie ein neues Terminal und führen Sie den wscat Befehl mit den folgenden Parametern aus.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Verwenden Sie den folgenden Befehl, um Ihre API zu testen.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId":"ABCD1=234",
 "requestId":"EFGH="}
```

API Gateway schlägt die Anfrage fehl.

Verwenden Sie den nächsten Befehl, um eine gültige Anfrage an Ihre API zu senden.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

Datentransformationen für WebSocket APIs einrichten

In API Gateway kann die Methodenanforderung einer WebSocket API eine Nutzlast in einem anderen Format als die entsprechende Nutzlast der Integrationsanfrage annehmen, wie es im Backend erforderlich ist. Das Backend wiederum kann eine Integrationsantwortnutzlast zurückgeben, die nicht der Methodenantwortnutzlast entspricht, die das Frontend erwartet.

In Amazon API Gateway können Sie Zuweisungsvorlagen verwenden, um die Nutzlast einer Methodenanforderung der entsprechenden Integrationsanforderung zuzuordnen bzw. die Nutzlast einer Integrationsanforderung der entsprechenden Methodenantwort zuzuordnen. Sie geben einen Vorlagen-Auswahlausdruck an, um zu bestimmen, welche Vorlage für die erforderlichen Datentransformationen verwendet werden soll.

Sie können Datenzuordnungen verwenden, um Daten aus einer Routenanforderung einer Backend-Integration zuzuordnen. Weitere Informationen hierzu finden Sie unter the section called “Datenzuordnung”.

Zuweisungsvorlagen und Modelle

Eine Zuweisungsvorlage ist ein in Velocity Template Language (VTL) ausgedrücktes Skript, dass mithilfe von JSONPath-Ausdrücken auf die Nutzlast angewendet wird. Weitere Informationen über API Gateway-Zuordnungsvorlagen finden Sie unter Grundlegendes zu Zuweisungsvorlagen.

Die Nutzlast kann ein dem JSON-Schema Entwurf 4 entsprechendes Datenmodell umfassen. Sie müssen kein Modell definieren, um eine Zuweisungsvorlage zu erstellen. Ein Modell kann Ihnen jedoch bei der Erstellung einer Vorlage helfen, da API Gateway einen Vorlagenentwurf auf der Grundlage eines bereitgestellten Modells generiert. Weitere Informationen über API Gateway-Modelle finden Sie unter Datenmodelle.

Vorlagen-Auswahlausdrücke

Um eine Nutzlast mit einer Zuordnungsvorlage zu transformieren, geben Sie in einer Integrationsanfrage oder Integrationsantwort einen Ausdruck für die WebSocket API-Vorlagenauswahl an. Durch Auswerten dieses Ausdrucks wird die Eingabe- oder Ausgabevorlage (falls vorhanden) bestimmt, die zum Transformieren des Anforderungstexts in den Integrationsanforderungstext (über eine Eingabevorlage) oder des Integrationsantworttext in den Routenantworttext (über eine Ausgabevorlage) verwendet werden soll.

`Integration.TemplateSelectionExpression` unterstützt `${request.body.jsonPath}` und statische Werte.

`IntegrationResponse.TemplateSelectionExpression` unterstützt `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` und statische Werte.

Integrationsantwort-Auswahlausdrücke

Wenn Sie eine Integrationsantwort für eine WebSocket API einrichten, können Sie optional einen Auswahlausdruck für eine Integrationsantwort angeben. Dieser Ausdruck bestimmt, welche `IntegrationResponse` ausgewählt werden sollte, wenn eine Integration zurückgegeben wird. Der Wert dieses Ausdrucks wird zurzeit durch API Gateway eingeschränkt, wie nachfolgend definiert. Beachten Sie, dass dieser Ausdruck nur für Nicht-Proxy-Integrationen relevant ist. Eine Proxy-Integration übergibt die Antwortnutzlast einfach ohne Modellierung oder Modifikation an die aufrufende Methode zurück.

Im Gegensatz zu den anderen vorhergehenden Auswahlausdrücken unterstützt dieser Ausdruck derzeit ein Musterabgleich-Format. Der Ausdruck sollte in Schrägstriche verpackt werden.

Derzeit ist der Wert je nach fixier `integrationType`:

- Für Lambda-basierte Integrationen lautet er `$integration.response.body.errorMessage`.
- Für `HTTP`- und `MOCK`-Integrationen ist dies `$integration.response.statuscode`.
- Für `HTTP_PROXY` und `AWS_PROXY` wird der Ausdruck nicht verwendet, da Sie anfordern, dass die Nutzlast über die aufrufende Methode übergeben wird.

Datenmapping für WebSocket APIs einrichten

Mit der Datenzuordnung können Sie Daten aus einer Routenanforderung einer Backend-Integration zuordnen.

> **Note**
>
> Die Datenzuordnung für WebSocket APIs wird in der nicht unterstützt AWS Management Console. Sie müssen das AWS CLI, oder ein SDK verwenden AWS CloudFormation, um die Datenzuordnung zu konfigurieren.

Themen

Zuordnen von Routenanforderungsdaten zu Integrationsanforderungsparametern

Integrationsanforderungsparameter können aus beliebigen definierten Routenanforderungsparametern, dem Anforderungstext `context` oder `stage`-Variablen und statischen Werten zugeordnet werden.

In der folgenden Tabelle ist *`PARAM_NAME`* der Name eines Routenanforderungsparameters des angegebenen Parametertyps. Er muss dem regulären Ausdruck entspreche `'^[a-zA-Z0-9._$-]+$]'`. *`JSONPath_Expression`* ist ein JSONPath-Ausdruck für ein JSON-Feld des Anforderungstests.

Zuweisungsausdrücke für Integrationsanforderungsdaten

Zugewiesene Datenquelle	Zuweisungsausdruck
Anforderungsabfragezeichenfolge (wird nur für die `$connect`-Route unterstützt)	`route.request.querystring.` *`PARAM_NAME`*
Anforderungs-Header (wird nur für die `$connect`-Route unterstützt)	`route.request.header.` *`PARAM_NAME`*
Abfragezeichenfolge mit mehreren Werten (nur für die `$connect`-Route unterstützt)	`route.request.multivaluequerystring.` *`PARAM_NAME`*
Anforderungsheader mit mehreren Werten (nur für die `$connect`-Route unterstützt)	`route.request.multivalueheader.` *`PARAM_NAME`*
Anforderungstext	`route.request.body.` *`JSONPath_EXPRESSION`*
Stufenvariablen	`stageVariables.` *`VARIABLE_NAME`*
Kontextvariablen	`context.`*`VARIABLE_NAME`* , wobei die Variable zu den unterstützten Kontextvariablen gehören muss.

Zugewiesene Datenquelle	Zuweisungsausdruck
Statischer Wert	*'STATIC_VALUE'* . *STATIC_VALUE* ist ein Zeichenfolgenliteral, das in einfache Anführungszeichen eingeschlossen werden muss.

Beispiele

In den folgenden AWS CLI Beispielen werden Datenzuordnungen konfiguriert. Eine AWS CloudFormation Beispielvorlage finden Sie unter. [websocket-data-mapping.yaml]

Zuordnen der ConnectionID eines Clients zu einem Header in einer Integrationsanforderung

Der folgende Beispielbefehl ordnet die `connectionId` eines Clients einem `connectionId`-Header in der Anforderung an eine Backend-Integration zu.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Zuordnen eines Abfragezeichenfolgenparameters zu einem Header in einer Integrationsanforderung

Die folgenden Beispielbefehle ordnen einen `authToken`-Abfragezeichenfolgenparameter einem `authToken`-Header in der Integrationsanforderung zu.

Fügen Sie zunächst den `authToken`-Abfragezeichenfolgenparameter zu den Anforderungsparametern der Route hinzu.

```
aws apigatewayv2 update-route --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameters '{"route.request.querystring.authToken": {"Required": false}}'
```

Ordnen Sie als Nächstes den Abfragezeichenfolgenparameter dem `authToken`-Header in der Anforderung der Backend-Integration zu.

```
aws apigatewayv2 update-integration \
```

```
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.authToken'='route.request.querystring.authToken'
```

Löschen Sie gegebenenfalls den `authToken`-Abfragezeichenfolgenparameter aus den Anforderungsparametern der Route.

```
aws apigatewayv2 delete-route-request-parameter \
    --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameter-key 'route.request.querystring.authToken'
```

Referenz API Gateway WebSocket API-Gateway-API-Zuordnungsvorlage

In diesem Abschnitt werden die Variablen zusammengefasst, die derzeit für WebSocket APIs in API Gateway unterstützt werden.

Parameter	Beschreibung
`$context.connectionId`	Eine eindeutige ID für die Verbindung, die für einen Rückruf an den Client verwendet werden kann.
`$context.connectedAt`	Die Epoch-formatierte Verbindungszeit.
`$context.domainName`	Ein Domainname für die WebSocket API. Dies kann für einen Rückruf an den Client (anstelle eines hardcodierten Wertes) verwendet werden.
`$context.eventType`	Der Ereignistyp: `CONNECT`, `MESSAGE` oder `DISCONNECT` .
`$context.messageId`	Eine eindeutige serverseitige ID für eine Nachricht. Nur verfügbar, wenn der `$context.eventType` `MESSAGE` ist.
`$context.routeKey`	Der ausgewählte Routenschlüssel.

Parameter	Beschreibung
`$context.requestId`	Entspricht `$context.extendedRequestId` .
`$context.extendedRequestId`	Eine automatisch generierte ID für den API-Aufruf, die weitere nützliche Informationen für das Debugging/die Fehlerbehebung enthält.
`$context.apiId`	Die ID, die API Gateway Ihrer API zuweist.
`$context.authorizer.principalId`	Die ID des Prinzipalbenutzer, die dem vom Client gesendeten und von einer Lambda-Funktion eines API Gateway-Lambda-Genehmigers (früher "benutzerdefinierter Genehmiger") zurückgegebenen Token zugeordnet ist.

Parameter	Beschreibung
`$context.authorizer.` *`property`*	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der `context`-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende `context`-Zuweisung zurück: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` Dann gibt der Aufruf von `$context.authorizer.key` die Zeichenfolge `"value"` zurück, der Aufruf von `$context.authorizer.numKey` gibt die Zeichenfolge `"1"` zurück und der Aufruf von `$context.authorizer.boolKey` gibt den Wert `"true"` zurück.
`$context.error.messageString`	Die Wert von `$context.error.message` in Anführungszeichen, d. h. `"$context.error.message"` .
`$context.error.validationErrorString`	Eine Zeichenfolge mit einer detaillierten Validierungs-Fehlermeldung.
`$context.identity.accountId`	Die der Anfrage zugeordnete AWS Konto-ID.
`$context.identity.apiKey`	Der API-Besitzerschlüssel, der der schlüsselfähigen API-Anforderung zugewiesen ist.
`$context.identity.apiKeyId`	Die API-Schlüssel-ID, die der schlüsselfähigen API-Anforderung zugewiesen ist

Parameter	Beschreibung
`$context.identity.caller`	Die Prinzipal-ID des Aufrufers, von dem die Anforderung stammt.
`$context.identity.cognitoAuthenticationProvider`	Eine durch Komma getrennte Liste der Amazon Cognito-Authentifizierungsanbieter, die vom Aufrufer, der die Anfrage stellt, verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Weitere Informationen finden Sie unter Verbundidentitäten verwenden im Amazon Cognito-Entwicklerhandbuch.
`$context.identity.cognitoAuthenticationType`	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind `authenticated` für authentifizierte Identitäten und `unauthenticated` für nicht authentifizierte Identitäten.
`$context.identity.cognitoIdentityId`	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Beschreibung
`$context.identity.cognitoId entityPoolId`	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
`$context.identity.sourceIp`	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
`$context.identity.user`	Die Prinzipal-ID des Benutzers, von dem die Anforderung stammt.
`$context.identity.userAgent`	Der Benutzeragent des API-Aufrufers.
`$context.identity.userArn`	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentif izierung.
`$context.requestTime`	Die Anforderungszeit im CLF-Format (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Die Anforderungszeit im Epoch-Format in Millisekunden.
`$context.stage`	Die Bereitstellungsstufe des API-Aufrufs (z. B. "Beta" oder "Prod").
`$context.status`	Der Antwortstatus.
`$input.body`	Gibt die Raw-Nutzlast als Zeichenfolge zurück.

Parameter	Beschreibung
`$input.json(x)`	Diese Funktion wertet einen JSONPath-Ausdruck aus und gibt die Ergebnisse als JSON-Zeichenfolge zurück. Beispielsweise gibt `$input.json('$.pets')` eine JSON-Zeichenfolge zurück, die die "Pets"-Struktur abbildet. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.

Parameter	Beschreibung
`$input.path(x)`	Empfängt eine JSONPath-Ausdruckszeichenfolge (x) und gibt eine JSON-Objektdarstellung des Ergebnisses zurück. Dies ermöglicht einen nativen Zugriff auf Elemente der Nutzlast in Apache Velocity Template Language (VTL) und deren Bearbeitung. Beispiel: Der Ausdruck `$input.path('$.pets')` könnte das folgende Objekt zurückgeben: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` gibt `"3"` zurück. Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.
`$stageVariables.` *`<variable_name>`*	*`<variable_name>`* variable name gibt einen Stufenvariablennamen an.

Parameter	Beschreibung
`$stageVariables['` *`<variable_name>`* `']`	*`<variable_name>`* repräsentiert einen beliebigen Stufenvariablennamen.
`${stageVariables['` *`<variable_name>`* `']}`	*`<variable_name>`* repräsentiert einen beliebigen Stufenvariablennamen.
`$util.escapeJavaScript()`	Escapiert die Zeichen in einer Zeichenfolge mithilfe von JavaScript Zeichenfolgenregeln. **Note** Mit dieser Funktion werden alle einfachen Anführungszeichen (`'`) durch Escape-Zeichen (`\'`) geschützt. Allerdings sind diese durch Escape-Zeichen geschützten einfachen Anführungszeichen in JSON nicht zulässig. Sofern die Ausgabe dieser Funktion in einer JSON-Eigenschaft verwendet werden soll, müssen alle einfachen Anführungszeichen, die durch Escape-Zeichen geschützt sind (`\'`), wieder in reguläre einfache Anführungszeichen (`'`) geändert werden. Das wird im folgenden Beispiel veranschaulicht: `$util.escapeJavaScript( data).replaceAll("\\'",",'")`

Parameter	Beschreibung

`$util.parseJson()`	Erhält das "stringify"-JSON-Objekt und gibt eine Objektdarstellung des Ergebnisses zurück. Mit dem Ergebnis dieser Funktion können Sie Elemente der Nutzlast, die in Apache Velocity Template Language (VTL) sind, aufrufen und bearbeiten. Angenommen, Sie haben folgende Nutzlast:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	Und verwenden die folgende Mapping-Vorlage:
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Dann erhalten Sie die folgende Ausgabe:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Konvertiert eine Zeichenfolge in das Format „application/x-www-form-urlencoded“.
`$util.urlDecode()`	Dekodiert eine Zeichenfolge „application/x-www-form-urlencoded“.
`$util.base64Encode()`	Codiert die Daten in eine base64-verschlüsselte Zeichenfolge.

Parameter	Beschreibung
`$util.base64Decode()`	Decodiert die Daten einer base64-verschlüsselten Zeichenfolge.

Arbeiten mit binären Medientypen für WebSocket APIs

WebSocket API-Gateway-APIs unterstützen derzeit keine binären Frames in Nutzlasten für eingehende Nachrichten. Wenn eine Client-Anwendung einen Binär-Frame sendet, weist API Gateway diesen zurück und trennt die Verbindung zum Client mit dem Code 1003.

Es gibt eine Problemumgehung für dieses Verhalten. Wenn der Client textcodierte binäre Daten (z. B. base64) als Text-Frame sendet, können Sie für die `contentHandlingStrategy`-Eigenschaft der Integration `CONVERT_TO_BINARY` festlegen, um die Nutzlast von einer base64-codierten Zeichenfolge in eine binäre Zeichenfolge zu konvertieren.

Damit für eine binäre Nutzlast in Nicht-Proxy-Integrationen eine Routenantwort zurückgegeben wird, können Sie für die `contentHandlingStrategy`-Eigenschaft der Integrationsantwort `CONVERT_TO_TEXT` festlegen, um die Nutzlast von einer binären in eine base64-codierte Zeichenfolge zu konvertieren.

Eine WebSocket API aufrufen

Nachdem Sie Ihre WebSocket API bereitgestellt haben, können Client-Anwendungen eine Verbindung zu ihr herstellen und Nachrichten an sie senden — und Ihr Backend-Service kann Nachrichten an verbundene Client-Anwendungen senden:

- Sie können `wscat` damit eine Verbindung zu Ihrer WebSocket API herstellen und Nachrichten an sie senden, um das Kundenverhalten zu simulieren. Siehe the section called “Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden”.
- Sie können die @connections-API aus Ihrem Backend-Service verwenden, um eine Rückrufnachricht an einen verbundenen Client zu senden, Verbindungsinformationen anzufordern oder die Clientverbindung zu trennen. Siehe the section called “Verwenden der `@connections`-Befehle in Ihrem Backend-Service”.
- Eine Client-Anwendung kann ihre eigene WebSocket Bibliothek verwenden, um Ihre WebSocket API aufzurufen.

Wird verwendet`wscat`, um eine Verbindung zu einer WebSocket API herzustellen und Nachrichten an diese zu senden

Das `wscat` Hilfsprogramm ist ein praktisches Tool zum Testen einer WebSocket API, die Sie in API Gateway erstellt und bereitgestellt haben. Sie können `wscat` wie folgt installieren und verwenden:

1. Laden Sie `wscat` von https://www.npmjs.com/package/wscat herunter.
2. Installieren Sie `wscat` mit dem folgenden Befehl:

   ```
   npm install -g wscat
   ```

3. Um eine Verbindung mit Ihrer API herzustellen, führen Sie wie im folgenden Beispiel gezeigt den Befehl `wscat` aus. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass die `Authorization`-Einstellung `NONE` lautet.

   ```
   wscat -c wss://aabbccddee.execute-api.us-east-1.amazonaws.com/test/
   ```

   Sie müssen *`aabbccddee`* durch die tatsächliche API-ID ersetzen, die in der API-Gateway-Konsole angezeigt oder vom AWS CLI -Befehl `create-api` zurückgegeben wird.

   Wenn sich Ihre API in einer anderen Region als `us-east-1` befindet, müssen Sie diese außerdem durch die korrekte Region ersetzen.

4. Zum Testen Ihrer API geben Sie eine Nachricht ähnlich der folgenden ein, während Sie noch verbunden sind:

   ```
   {"{jsonpath-expression}":"{route-key}"}
   ```

   wobei *`{jsonpath-expression}`* ein JSONPath-Ausdruck und *`{route-key}`* ein Routenschlüssel für die API ist. Zum Beispiel:

   ```
   {"action":"action1"}
   {"message":"test response body"}
   ```

   Weitere Informationen zu JSONPath finden Sie unter JSONPath oder JSONPath for Java.

5. Um die Verbindung mit der API zu trennen, geben Sie ei `ctrl-C`.

Verwenden der **`@connections`**-Befehle in Ihrem Backend-Service

Ihr Back-End-Dienst kann die folgenden WebSocket HTTP-Verbindungsanfragen verwenden, um eine Rückrufnachricht an einen verbundenen Client zu senden, Verbindungsinformationen abzurufen oder die Verbindung zum Client zu trennen.

> **Important**
>
> Da für diese Anfragen die IAM-Autorisierung erforderlich ist, müssen Sie sich mit Signature Version 4 (SigV4) anmelden. Dazu können Sie die API Gateway-Verwaltungs-API verwenden. Weitere Informationen finden Sie unter. ApiGatewayManagementApi

Im folgenden Befehl müssen Sie ihn durch die tatsächliche API-ID *{api-id}* ersetzen, die in der API Gateway Gateway-Konsole angezeigt oder vom Befehl AWS CLI create-api zurückgegeben wird. Die Verbindung muss hergestellt werden, bevor Sie diesen Befehl ausführen können.

Um eine Rückrufnachricht zum Client zu senden, verwenden Sie Folgendes:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Sie können diese Anforderung mithilfe von `Postman` oder durch Aufruf von `awscurl` wie im folgenden Beispiel testen:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Sie müssen den Befehl wie im folgenden Beispiel URL-codieren:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccddee.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Zum Abrufen des neuesten Verbindungsstatus des Clients verwenden Sie:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Zum Trennen der Verbindung mit dem Client verwenden Sie:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Sie können mittels der `$context`-Variablen in Ihrer Integration eine Rückruf-URL dynamisch erstellen. Wenn Sie zum Beispiel die Lambda-Proxy-Integration mit einer `Node.js`-Lambda-Funktion verwenden, können Sie die URL wie folgt aufbauen und eine Nachricht an einen verbundenen Client senden:

```
import {
  ApiGatewayManagementApiClient,
  PostToConnectionCommand,
} from "@aws-sdk/client-apigatewaymanagementapi";

export const handler = async (event) => {
  const domain = event.requestContext.domainName;
  const stage = event.requestContext.stage;
  const connectionId = event.requestContext.connectionId;
  const callbackUrl = `https://${domain}/${stage}`;
  const client = new ApiGatewayManagementApiClient({ endpoint: callbackUrl });

  const requestParams = {
    ConnectionId: connectionId,
    Data: "Hello!",
  };

  const command = new PostToConnectionCommand(requestParams);

  try {
    await client.send(command);
  } catch (error) {
    console.log(error);
  }

  return {
    statusCode: 200,
  };
};
```

Wenn Sie eine Rückrufnachricht senden, muss Ihre Lambda-Funktion über die Berechtigung verfügen, die API-Gateway-Management-API aufzurufen. Möglicherweise erhalten Sie eine

Fehlermeldung mit dem Inhalt `GoneException`, wenn Sie eine Nachricht veröffentlichen, bevor die Verbindung hergestellt wurde oder nachdem der Client die Verbindung getrennt hat.

WebSocket APIs veröffentlichen, die Kunden aufrufen können

Das einfache Erstellen und Entwickeln einer API Gateway-API macht sie nicht automatisch für Ihre Benutzer aufrufbar. Um sie aufrufbar zu machen, müssen Sie Ihre API in einer Stufe bereitstellen. Darüber hinaus können Sie die URL anpassen, die Ihre Benutzer für den Zugriff auf Ihre API verwenden. Sie können ihr eine Domäne zuweisen, die mit Ihrer Marke übereinstimmt oder einprägsamer ist als die Standard-URL für Ihre API.

In diesem Abschnitt erfahren Sie, wie Sie Ihre API bereitstellen und die URL anpassen, die Sie Benutzern für den Zugriff zur Verfügung stellen.

> **Note**
>
> Um die Sicherheit Ihrer API-Gateway-APIs zu erhöhen, ist die `execute-api.{region}.amazonaws.com`-Domain in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-`-Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domain-Namen für Ihre API-Gateway-APIs einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Themen

- Arbeiten mit Stufen für WebSocket APIs
- Stellen Sie eine WebSocket API in API Gateway bereit
- Sicherheitsrichtlinie für WebSocket APIs
- Einrichtung benutzerdefinierter Domainnamen für WebSocket APIs

Arbeiten mit Stufen für WebSocket APIs

Eine API-Stufe ist ein logischer Verweis auf einen Lebenszyklusstatus Ihrer API (z. B. `dev`, `prod`, `beta` oder `v2`). API-Stufen werden durch ihre API-ID und den Stufennamen identifiziert und sind in

der URL enthalten, die Sie zum Aufrufen der API verwenden. Jede Stufe ist ein benannter Verweis auf eine Bereitstellung der API und wird für Clientanwendungen zum Aufrufen zur Verfügung gestellt.

Eine Bereitstellung ist ein Snapshot Ihrer API-Konfiguration. Nachdem Sie eine API in einer Phase bereitgestellt haben, kann sie von Clients aufgerufen werden. Sie müssen eine API bereitstellen, damit Änderungen wirksam werden.

Stufenvariablen

Stufenvariablen sind Schlüssel-Wert-Paare, die Sie für eine Phase einer API definieren können. WebSocket Sie weisen dasselbe Verhalten auf wie Umgebungsvariablen und können für die API-Einrichtung verwendet werden.

Sie können beispielsweise eine Stufenvariable definieren und dann ihren Wert als HTTP-Endpunkt für eine HTTP-Proxy-Integration festlegen. Später können Sie den Endpunkt mithilfe des zugeordneten Stufenvariablennamens referenzieren. Auf diese Weise können Sie in jeder Stufe dieselbe API-Einrichtung mit einem anderen Endpunkt verwenden. In ähnlicher Weise können Sie Stufenvariablen verwenden, um für jede Phase Ihrer API eine andere AWS Lambda Funktionsintegration anzugeben.

> **Note**
>
> Stage-Variablen sind nicht dazu gedacht, für sensible Daten wie Anmeldeinformationen verwendet zu werden. Verwenden Sie einen AWS Lambda Autorisierer, um sensible Daten an Integrationen weiterzugeben. Sie können sensible Daten an Integrationen in der Ausgabe des Lambda-Genehmigers übergeben. Weitere Informationen hierzu finden Sie unter the section called “Antwortformat des Lambda-Genehmigers”.

Beispiele

Wenn Sie eine Stufenvariable zum Anpassen des HTTP-Integrationsendpunkts verwenden möchten, müssen Sie zunächst den Namen und den Wert der Stufenvariablen (z. B. `url`) mit dem Wert `example.com` festlegen. Richten Sie als Nächstes eine HTTP-Proxy-Integration ein. Anstatt die URL des Endpunkts einzugeben, können Sie API Gateway anweisen, den Wert der Stufenvariablen zu verwenden, **`http://${stageVariables.url}`**. Dieser Wert weist API Gateway an, Ihre Stufenvariable `${}` zur Laufzeit abhängig von der Stufe Ihrer API zu ersetzen.

Sie können auf ähnliche Weise auf Stufenvariablen verweisen, um einen Lambda-Funktionsnamen oder einen AWS Rollen-ARN anzugeben.

Wenn Sie einen Lambda-Funktionsnamen als Stufenvariablenwert angeben, müssen Sie die Berechtigungen für die Lambda-Funktion manuell konfigurieren. Dafür können Sie die AWS Command Line Interface (AWS CLI) verwenden.

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

API Gateway Stufenvariablenreferenz

HTTP-Integrations-URIs

Sie können eine Stufenvariable als Teil einer HTTP-Integrations-URI verwenden, wie in den folgenden Beispielen gezeigt.

- Eine vollständige URI ohne Protokoll – `http://${stageVariables.<variable_name>}`
- Eine vollständige Domäne – `http://${stageVariables.<variable_name>}/resource/operation`
- Eine Unterdomäne – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Ein Pfad – `http://example.com/${stageVariables.<variable_name>}/bar`
- Eine Abfragezeichenfolge – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Lambda-Funktionen

Sie können eine Stufenvariable anstelle eines Lambda-Funktionsnamens oder Alias verwenden, wie in den folgenden Beispielen gezeigt.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Um eine Stufenvariable für eine Lambda-Funktion zu verwenden, muss sich die Funktion im selben Konto wie die API befinden. Stufenvariablen unterstützen keine kontoübergreifenden Lambda-Funktionen.

AWS Anmeldeinformationen für die Integration

Sie können eine Stage-Variable als Teil eines ARN mit AWS Benutzer- oder Rollenanmeldedaten verwenden, wie im folgenden Beispiel gezeigt.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Stellen Sie eine WebSocket API in API Gateway bereit

Nachdem Sie Ihre WebSocket API erstellt haben, müssen Sie sie bereitstellen, damit Ihre Benutzer sie aufrufen können.

Zum Bereitstellen einer API erstellen Sie eine API-Bereitstellung und verknüpfen sie mit einer Stufe. Jede Stufe ist ein Snapshot der API und wird für Client-Apps zum Aufrufen verfügbar gemacht.

> **Important**
>
> Eine API muss nach jeder Aktualisierung erneut bereitgestellt werden. Alle Änderungen außer an Stufeneinstellungen erfordern eine erneute Bereitstellung, einschließlich Änderungen an den folgenden Ressourcen:
>
> - Routen
> - Integrationen
> - Authorizers
>
> APIs sind standardmäßig auf 10 Stufen begrenzt. Wir empfehlen, bereits verwendete Stufen für Ihre Bereitstellungen wiederzuverwenden.

Um eine bereitgestellte WebSocket API aufzurufen, sendet der Client eine Nachricht an die URL der API. Die URL wird anhand des Hostnamens und der Stufennummer der API bestimmt.

> **Note**
>
> API Gateway unterstützt Nutzlasten bis zu 128 KB mit einer maximalen Framegröße von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind.

Unter Verwendung des Standard-Domainnamens der API hat die URL (zum Beispiel) einer WebSocket API in einer bestimmten Phase (*{stageName}*) das folgende Format:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Um die URL der WebSocket API benutzerfreundlicher zu gestalten, können Sie einen benutzerdefinierten Domainnamen (z. B.`api.example.com`) erstellen, der den Standard-Hostnamen der API ersetzt. Der Konfigurationsvorgang ist bei REST-APIs identisch. Weitere Informationen finden Sie unter the section called “Benutzerdefinierte Domänennamen”.

Stufen ermöglichen eine robuste Versionskontrolle Ihrer API. Sie können beispielsweise eine API für eine `test`- und eine `prod`-Stufe bereitstellen und die `test`-Stufe als Test-Build und die `prod`-Stufe als stabilen Build verwenden. Nachdem die Aktualisierungen den Test bestanden haben, können Sie die `test`-Stufe auf `prod` hochstufen. Die Hochstufung kann durch die erneute Bereitstellung der API für die `prod`-Stufe erfolgen. Weitere Informationen zu Stufen finden Sie unter the section called “Einrichten einer Stufe”.

Themen

- Erstellen Sie eine WebSocket API-Bereitstellung mit dem AWS CLI
- Erstellen Sie eine WebSocket API-Bereitstellung mit der API Gateway Gateway-Konsole

Erstellen Sie eine WebSocket API-Bereitstellung mit dem AWS CLI

Um eine Bereitstellung AWS CLI zu erstellen, verwenden Sie den Befehl create-deployment, wie im folgenden Beispiel gezeigt:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccddee
```

Beispielausgabe:

```
{
```

```
    "DeploymentId": "fedcba",
    "DeploymentStatus": "DEPLOYED",
    "CreatedDate": "2018-11-15T06:49:09Z"
}
```

Die bereitgestellte API kann erst aufgerufen werden, wenn Sie die Bereitstellung einer Stufe zugeordnet haben. Sie können eine neue Stufe erstellen oder eine Stufe wiederverwenden, die Sie zuvor erstellt haben.

Um eine neue Phase zu erstellen und sie der Bereitstellung zuzuordnen, verwenden Sie den Befehl create-stage, wie im folgenden Beispiel gezeigt:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccddee --deployment-id
 fedcba --stage-name test
```

Beispielausgabe:

```
{
    "StageName": "test",
    "CreatedDate": "2018-11-15T06:50:28Z",
    "DeploymentId": "fedcba",
    "DefaultRouteSettings": {
        "MetricsEnabled": false,
        "ThrottlingBurstLimit": 5000,
        "DataTraceEnabled": false,
        "ThrottlingRateLimit": 10000.0
    },
    "LastUpdatedDate": "2018-11-15T06:50:28Z",
    "StageVariables": {},
    "RouteSettings": {}
}
```

Um eine vorhandene Phase wiederzuverwenden, aktualisieren Sie die `deploymentId` Eigenschaft der Phase mit der neu erstellten Bereitstellungs-ID (*`{deployment-id}`*), indem Sie den Befehl update-stage verwenden.

```
 aws apigatewayv2 update-stage --region {region} \
    --api-id {api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id}
```

Erstellen Sie eine WebSocket API-Bereitstellung mit der API Gateway Gateway-Konsole

So verwenden Sie die API Gateway Gateway-Konsole, um eine Bereitstellung für eine WebSocket API zu erstellen:

1. Melden Sie sich bei der API Gateway-Konsole an und wählen Sie die API aus.
2. Klicken Sie auf Deploy API.
3. Wählen Sie die gewünschte Stufe aus der Dropdown-Liste aus oder geben Sie den Namen einer neuen Stufe ein.

Sicherheitsrichtlinie für WebSocket APIs

API Gateway setzt eine Sicherheitsrichtlinie von `TLS_1_2` für alle WebSocket API-Endpunkte durch.

Eine Sicherheitsrichtlinie ist eine vordefinierte Kombination aus TLS-Mindestversion und Verschlüsselungssammlungen, die von Amazon API Gateway angeboten werden. Das TLS-Protokoll behandelt Netzwerksicherheitsprobleme wie Manipulationen und Abhören zwischen einem Client und einem Server. Wenn Ihre Clients über die benutzerdefinierte Domäne einen TLS-Handshake mit Ihrer API ausführen, erzwingt die Sicherheitsrichtlinie die TLS-Version und die Verschlüsselungssuite-Optionen, die von Ihren Clients verwendet werden können. Diese Sicherheitsrichtlinie akzeptiert TLS 1.2- und TLS 1.3-Verkehr und lehnt TLS 1.0-Verkehr ab.

Unterstützte TLS-Protokolle und Chiffren für APIs WebSocket

In der folgenden Tabelle werden die unterstützten TLS-Protokolle und Chiffren für APIs beschrieben. WebSocket

Sicherheitsrichtlinie	TLS_1_2
TLS-Protokolle	
TLSv1.3	♦
TLSv1.2	♦
TLS-Chiffren	

Sicherheitsrichtlinie	TLS_1_2
TLS_AES_128_GCM_SHA256	♦
TLS_AES_256_GCM_SHA384	♦
TLS_CHACHA20_POLY1305_SHA256	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

OpenSSL- und RFC-Verschlüsselungsnamen

OpenSSL und IETF RFC 5246 verwenden unterschiedliche Namen für dieselben Chiffren. Eine Liste der Chiffriernamen finden Sie unter. the section called “OpenSSL- und RFC-Verschlüsselungsnamen”

Informationen zu REST-APIs und HTTP-APIs

Weitere Informationen zu REST-APIs und HTTP-APIs finden Sie unter the section called “Auswahl einer Sicherheitsrichtlinie” undthe section called “Sicherheitsrichtlinie für HTTP-APIs”.

Einrichtung benutzerdefinierter Domainnamen für WebSocket APIs

Benutzerdefinierte Domänennamen sind einfachere und intuitivere URLs, die Sie Ihren API-Benutzern zur Verfügung stellen können.

Nach der Bereitstellung der API können Sie (und Ihre Kunden) die API mit der Standardstamm-URL im folgenden Format aufrufen:

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

wo von API Gateway generiert `api-id` wird, `region` (AWS Region) wird von Ihnen bei der Erstellung der API angegeben und `stage` wird von Ihnen bei der Bereitstellung der API angegeben.

Der Hostname-Teil der URL (d. h. *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`) verweist auf einen API-Endpunkt. Der Standard-API-Endpunkt hat möglicherweise einen schwer zu merkenden, wenig benutzerfreundlichen Namen.

Mit benutzerdefinierten Domänennamen können Sie den Hostnamen Ihrer API einrichten und einen Basispfad (z. B. `myservice`) auswählen, um die alternative URL Ihrer API zuzuordnen. Eine benutzerfreundlichere API-Basis-URL kann dann folgendermaßen aussehen:

```
https://api.example.com/myservice
```

> **Note**
>
> Ein benutzerdefinierter Domainname für eine WebSocket API kann nicht REST-APIs oder HTTP-APIs zugeordnet werden.
> Für WebSocket APIs werden regionale benutzerdefinierte Domainnamen unterstützt.
> Für WebSocket APIs ist TLS 1.2 die einzige unterstützte TLS-Version.

Registrieren eines Domänennamens

Wenn Sie benutzerdefinierte Domänennamen für Ihre APIs einrichten möchten, müssen Sie eine Internetdomäne registrieren. Ihr Domainname muss der RFC 1035-Spezifikation entsprechen und darf ein Maximum von 63 Cts pro Label und 255 Octeks insgesamt haben. Falls erforderlich, können Sie eine Internetdomäne mit Amazon Route 53 oder mit einem Domain-Registrar Ihrer Wahl registrieren. Der benutzerdefinierte Domänenname einer API kann dem Namen einer Unter- oder Stammdomäne (auch als „Zone Apex“ bezeichnet) einer registrierten Internetdomäne entsprechen.

Nachdem ein benutzerdefinierter Domänenname in API Gateway erstellt wurde, müssen Sie den Ressourceneintrag Ihres DNS-Providers erstellen oder aktualisieren, um ihn Ihrem API-Endpunkt zuzuordnen. Ohne eine solche Zuordnung können API-Anfragen für den benutzerdefinierten Domänennamen API Gateway nicht erreichen.

Regionale benutzerdefinierte Domänennamen

Wenn Sie einen benutzerdefinierten Domänennamen für eine regionale API erstellen, erstellt API Gateway einen regionalen Domänennamen für die API. Sie müssen einen DNS-Datensatz so einrichten, dass der benutzerdefinierte Domänenname dem regionalen Domänennamen zugeordnet wird. Außerdem müssen Sie ein Zertifikat für den benutzerdefinierten Domänennamen bereitstellen.

Benutzerdefinierte Domänennamen mit Platzhalter

Mit benutzerdefinierten Platzhalter-Domänennamen können Sie eine nahezu unendliche Anzahl von Domänennamen unterstützen, ohne das Standardkontingentzu überschreiten. Zum Beispiel könnten Sie jedem Ihrer Kunden einen eigenen Domänennamen geben, *`customername`*`.api.example.com`.

Um einen benutzerdefinierten Platzhalterdomänennamen zu erstellen, geben Sie einen Platzhalter (`*`) als erste Subdomäne einer benutzerdefinierten Domäne an, die alle möglichen Subdomänen einer Root-Domäne darstellt.

Der benutzerdefinierte Domänenname mit Platzhalter `*.example.com` führt beispielsweise zu Unterdomänen wie `a.example.com`, `b.example.com` und `c.example.com`, die alle zur gleichen Domäne weiterleiten.

Benutzerdefinierte Domänennamen mit Platzhaltern unterstützen andere Konfigurationen als die benutzerdefinierten Standarddomänennamen von API Gateway. In einem einzigen AWS Konto können Sie beispielsweise unterschiedliche Einstellungen vornehmen `*.example.com` und `a.example.com` sich unterschiedlich verhalten.

Sie können die Kontextvariablen `$context.domainName` und `$context.domainPrefix` verwenden, um den Domänennamen zu bestimmen, mit dem ein Client Ihre API aufruft. Weitere Informationen zu Kontextvariablen finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen.

Um einen benutzerdefinierten Domänennamen mit Platzhalter zu erstellen, müssen Sie ein von ACM ausgestelltes Zertifikat angeben, das mithilfe der DNS- oder der E-Mail-Validierungsmethode validiert wurde.

> **Note**
>
> Sie können keinen benutzerdefinierten Platzhalter-Domänennamen erstellen, wenn ein anderes AWS Konto einen benutzerdefinierten Domänennamen erstellt hat, der mit dem benutzerdefinierten Platzhalter-Domänennamen in Konflikt steht. Wurde `a.example.com` beispielsweise von Konto A erstellt, dann kann Konto B als benutzerdefinierten Domänennamen mit Platzhalter nicht `*.example.com` erstellen.
> Wenn Konto A und Konto B den gleichen Besitzer haben, können Sie sich an das AWS Supportcenter wenden, um eine Ausnahme anzufordern.

Zertifikate für benutzerdefinierte Domänennamen

> **Important**
>
> Sie geben das Zertifikat für Ihren benutzerdefinierten Domänennamen an. Wenn Ihre Anwendung Zertifikat-Pinning, manchmal auch SSL-Pinning genannt, verwendet, um ein ACM-Zertifikat anzuheften, kann die Anwendung nach der Verlängerung des Zertifikats möglicherweise keine Verbindung zu Ihrer Domain herstellen. AWS Weitere Informationen finden Sie unter Probleme beim Zertifikats-Pinning im AWS Certificate Manager - Benutzerhandbuch.

Um ein Zertifikat für einen benutzerdefinierten Domänennamen in einer Region bereitzustellen, in der ACM unterstützt wird, müssen Sie ein Zertifikat von ACM anfordern. Um ein Zertifikat für einen regionalen benutzerdefinierten Domänennamen in einer von ACM nicht unterstützten Region bereitzustellen, müssen Sie ein Zertifikat für API Gateway in dieser Region importieren.

Wenn Sie ein SSL-/TLS-Zertifikat importieren möchten, müssen Sie den PEM-formatierten Hauptteil des SSL-/TLS-Zertifikats, den persönlichen Schlüssel und die Zertifikatkette für den benutzerdefinierten Domänennamen bereitstellen. Ein in ACM gespeichertes Zertifikat wird durch seinen ARN angegeben. Um ein AWS verwaltetes Zertifikat für einen Domainnamen zu verwenden, verweisen Sie einfach auf dessen ARN.

Die Einrichtung und Verwendung eines benutzerdefinierten Domänennamens für eine API in ACM ist ganz einfach. Sie erstellen ein Zertifikat für den angegebenen Domänennamen (oder importieren ein Zertifikat), richten den Domänennamen in API Gateway mit dem ARN des von ACM bereitgestellten Zertifikats ein und ordnen einen Basispfad unter dem benutzerdefinierten Domänennamen einer

Bereitstellungsphase der API zu. Wenn Sie Zertifikate verwenden, die von ACM ausgestellt werden, brauchen sich keine Gedanken über vertrauliche Zertifikatdetails wie private Schlüssel zu machen.

Einen benutzerdefinierten Domain-Namen einrichten

Weitere Informationen zum Einrichten eines benutzerdefinierten Domänennamens finden Sie unter Vorbereiten von Zertifikaten in AWS Certificate Manager und Einrichten eines regionalen benutzerdefinierten Domänennamens in API Gateway.

Arbeiten mit API-Mappings für APIs WebSocket

Sie verwenden API-Mappings, um API-Stufen mit einem benutzerdefinierten Domain-Namen zu verbinden. Nachdem Sie einen Domain-Namen erstellt und DNS-Einträge konfiguriert haben, verwenden Sie API-Mappings, um Datenverkehr über Ihren benutzerdefinierten Domain-Namen an Ihre APIs zu senden.

Ein API-Mapping gibt eine API, eine Phase und optional einen Pfad an, die für das Mapping verwendet werden sollen. Sie können beispielsweise die `production`-Phase einer API in `wss://api.example.com/orders` abbilden.

Bevor Sie eine API-Zuweisung erstellen, benötigen Sie eine API, eine Phase und einen benutzerdefinierten Domain-Namen. Weitere Informationen zum Erstellen eines benutzerdefinierten Domain-Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

Einschränkungen

- Bei einer API-Zuordnung müssen sich der benutzerdefinierte Domainname und die zugewiesenen APIs im selben Konto befinden. AWS
- API-Mappings dürfen nur Buchstaben, Zahlen und die folgenden Zeichen enthalten: `$-_.+!*'()`.
- Die maximale Länge für den Pfad in eines API-Mappings beträgt 300 Zeichen.
- Sie können WebSocket APIs nicht demselben benutzerdefinierten Domainnamen zuordnen wie eine HTTP-API oder REST-API.

Ein API-Mapping erstellen

Um ein API-Mapping zu erstellen, müssen Sie zuerst einen benutzerdefinierten Domain-Namen, eine API und eine Phase erstellen. Informationen zum Erstellen eines benutzerdefinierten Domain-

Namens finden Sie unter the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.

AWS Management Console

So erstellen Sie eine API-Zuweisung

1. Melden Sie sich bei der API-Gateway-Konsole unter https://console.aws.amazon.com/apigateway an.
2. Wählen Sie benutzerdefinierte Domain-Namen aus.
3. Wählen Sie einen benutzerdefinierten Domain-Namen aus, den Sie bereits erstellt haben.
4. Wählen Sie API-Mappings aus.
5. Wählen Sie API-Zuordnungen konfigurieren aus.
6. Wählen Sie Neue Zuordnung hinzufügen aus.
7. Geben Sie eine API, eine Phase und optional einen Pfad ein.
8. Wählen Sie Save (Speichern) aus.

AWS CLI

Der folgende AWS CLI Befehl erstellt eine API-Zuordnung. In diesem Beispiel sendet API Gateway Anforderungen an `api.example.com/v1` an die angegebene API und Phase.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1 \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

Das folgende AWS CloudFormation Beispiel erstellt eine API-Zuordnung.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
```

```
    Stage: !Ref MyStage
```

Deaktivierung des Standardendpunkts für eine API WebSocket

Standardmäßig können Clients Ihre API mithilfe des `execute-api`-Endpunkts aufrufen, den API Gateway für Ihre API generiert. Um sicherzustellen, dass Kunden nur über einen benutzerdefinierten Domänennamen auf Ihre API zugreifen können, deaktivieren Sie den standardmäßigen `execute-api`-Endpunkt.

> **Note**
>
> Wenn Sie den Standardendpunkt deaktivieren, wirkt sich dies auf alle Stufen einer API aus.

Der folgende AWS CLI Befehl deaktiviert den Standardendpunkt für eine WebSocket API.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Nachdem Sie den Standardendpunkt deaktiviert haben, müssen Sie Ihre API bereitstellen, damit die Änderung wirksam wird.

Der folgende AWS CLI Befehl erstellt eine Bereitstellung.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Schützen Sie Ihre WebSocket API

Sie können Drosselungen für Ihre APIs konfigurieren, um sie davor zu schützen, von zu vielen Anfragen überfordert zu werden. Drosselungen werden mit bestem Bemühen angewendet und sollten als Ziele und nicht als garantierte Anforderungsobergrenzen betrachtet werden.

API Gateway drosselt Anfragen an Ihre API mit dem Token-Bucket-Algorithmus, wobei ein Token für eine Anforderung gilt. Insbesondere überprüft API Gateway die Rate und einen Burst von

Anfragen für alle APIs in Ihrem Konto pro Region. Im Token-Bucket-Algorithmus kann ein Burst ein vordefiniertes Überlaufen dieser Grenzwerte ermöglichen, aber andere Faktoren können auch dazu führen, dass Grenzwerte in einigen Fällen überlaufen werden.

Wenn Anfrageeinreichungen die Steady-State-Anfragerate und Steigerungs-Limits überschreiten, drosselt API Gateway Anfragen. Kunden erhalten möglicherweise `429 Too Many Requests` Fehlerantworten an dieser Stelle. Bei der Erfassung solcher Ausnahmen kann der Client die fehlgeschlagenen Anforderungen in einer Weise erneut einreichen, die raten-begrenzend ist.

Als API-Entwickler können Sie die Target-Limits für individuelle API-Stufen oder -Routen festlegen, um die Gesamtleistung in allen APIs in Ihrem Konto zu verbessern.

Drosselung auf Kontoebene pro Region

Standardmäßig begrenzt API Gateway die Steady-State-Anfragen pro Sekunde (RPS) für alle APIs innerhalb eines AWS -Kontos pro Region. Die Steigerung (also die maximale Bucket-Größe) wird über alle APIs innerhalb eines AWS -Kontos pro Region wird ebenfalls beschränkt. In API Gateway entspricht das Burst-Limit der maximalen Target-Anzahl gleichzeitiger Anfragen, die API Gateway vor Rückgabe von `429 Too Many Requests`-Fehlerantworten ausführt. Weitere Informationen zu Drosselungskontingenten finden Sie unter Kontingente und wichtige Hinweise.

Grenzwerte pro Konto werden auf alle APIs in einem Konto in einer bestimmten Region angewendet. Das Ratenlimit auf Kontoebene kann auf Anforderung erhöht werden – höhere Limits sind mit APIs möglich, die kürzere Timeouts und kleinere Nutzlasten aufweisen. Um eine Steigerung der Ablehnungslimits auf Kontoebene pro Region anzufordern, wenden Sie sich an das AWS Supportcenter. Weitere Informationen finden Sie unter Kontingente und wichtige Hinweise. Beachten Sie, dass diese Grenzwerte nicht höher als die AWS Drosselungsgrenzen sein können.

Drosselung auf Routenebene

Sie können die Drosselung auf Routenebene festlegen, um die Anforderungs-Drosselungslimits auf Kontoebene für eine bestimmte Stufe oder für individuelle Routen in Ihrer API zu überschreiben. Die Standardgrenzen für die Routendrosselung können die Ratenlimits auf Kontoebene nicht überschreiten.

Sie können die Drosselung auf Routenebene konfigurieren, indem Sie die verwende AWS CLI. Mit dem folgenden Befehl wird die benutzerdefinierte Einschränkung für die angegebene Stufe und Route einer API konfiguriert.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"messages":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

WebSocket APIs überwachen

Sie können CloudWatch Metriken und CloudWatch Protokolle verwenden, um WebSocket APIs zu überwachen. Durch die Kombination von Protokollen und Metriken können Sie Fehler protokollieren und die Leistung Ihrer API überwachen.

> **Note**
>
> API Gateway generiert in den folgenden Fällen möglicherweise keine Protokolle und Metriken:
>
> - 413 Request Entity Too Large-Fehler
> - Übermäßige 429 Too Many Requests-Fehler
> - Fehler der Serie 400 von Anfragen, die an eine benutzerdefinierte Domäne gesendet wurden, die keine API-Zuordnung hat
> - Fehler der Serie 500, die durch interne Ausfälle verursacht werden

Themen

Überwachung der WebSocket API-Ausführung mit CloudWatch Metriken

Sie können CloudWatchAmazon-Metriken verwenden, um WebSocket APIs zu überwachen. Die Konfiguration ist mit der vergleichbar, die für REST-APIs verwendet wird. Weitere Informationen finden Sie unter Überwachung der REST-API-Ausführung mit CloudWatch Amazon-Metriken.

Die folgenden Metriken werden für WebSocket APIs unterstützt:

Metrik	Beschreibung
ConnectCount	Die Anzahl der Nachrichten, die an die `$connect`-Routenintegration gesendet werden.
MessageCount	Die Anzahl der Nachrichten, die entweder vom oder an den Client an die WebSocket API gesendet wurden.
IntegrationError	Die Anzahl der Anforderungen, die eine 4XX/5XX-Antwort von der Integration zurückgeben.
ClientError	Die Anzahl der Anfragen, auf die eine 4XX-Antwort von API Gateway zurückgegeben wird, bevor die Integration aufgerufen wird.
ExecutionError	Fehler beim Aufrufen der Integration.
IntegrationLatency	Der Zeitunterschied zwischen dem Senden der Anfragen durch API Gateway an die Integration und dem Eingang der Antwort der Integration bei API Gateway. Wird für Rückrufe und Pseudo-Integrationen ausgeblendet.

Sie können die Dimensionen in der folgenden Tabelle verwenden, um API Gateway-Metriken zu filtern.

Dimension	Beschreibung
ApiId	Filtert API Gateway-M etriken für eine API mit der angegebenen API-ID.
ApiId, Phase	Filtert API Gateway-Metriken für eine API-Stufe mit der angegebenen API-ID und Stufen-ID.
ApiId, Methode, Ressource, Phase	Filtert API-Gateway-Metrik en für eine API-Methode mit der angegebenen API-ID, Stufen-ID, Ressourcenpfad und Routen-ID. API Gateway sendet diese Metriken nur, wenn Sie detaillierte CloudWatch Metriken explizit aktiviert haben. Sie können dies tun, indem Sie die UpdateSta geAktion der API Gateway V2 REST-API aufrufen, auf die die `detailedMetricsEna bled` Eigenschaft aktualisi ert `true` werden soll. Alternativ können Sie den AWS CLI Befehl update-stage aufrufen, um die `DetailedMetricsEna bled` Eigenschaft auf zu aktualisieren. `true` Durch die Aktivierung dieser Metriken wird Ihr Konto mit zusätzlichen Gebühren belastet. Preisinfo

Dimension	Beschreibung
	rmationen finden Sie unter CloudWatchAmazon-Preise.

Protokollierung für eine WebSocket API konfigurieren

Sie können die Protokollierung aktivieren, um CloudWatch Protokolle in Logs zu schreiben. Es gibt zwei Arten der API-Anmeldung CloudWatch: Ausführungsprotokollierung und Zugriffsprotokollierung. Bei der Ausführungsprotokollierung verwaltet API Gateway die CloudWatch Protokolle. Der Prozess umfasst das Erstellen von Protokollgruppen und -Streams sowie die Meldung der Anforderungen und Antworten jedes Aufrufers an Protokoll-Streams.

In der Zugriffsprotokollierung protokollieren Sie, als API-Entwickler, wer auf Ihre API zugegriffen hat und wie der Aufrufer auf die API zugegriffen hat. Erstellen Sie eine eigene Protokollgruppe oder wählen Sie eine vorhandene Protokollgruppe aus, die von API Gateway verwaltet werden kann. Um die Zugriffsdetails anzugeben, wählen Sie `$context`-Variablen (ausgedrückt in einem Format Ihrer Wahl) und eine Protokollgruppe als Ziel aus.

Anweisungen zum Einrichten der CloudWatch Protokollierung finden Sie unterthe section called “ CloudWatch API-Protokollierung mit der API Gateway Gateway-Konsole einrichten”.

Wenn Sie das Log Format (Protokollformat) angeben, können Sie wählen, welche Kontextvariablen protokolliert werden sollen. Die folgenden Variablen werden unterstützt.

Parameter	Beschreibung
`$context.apiId`	Die ID, die API Gateway Ihrer API zuweist.
`$context.authorize.error`	Die Autorisierungsfehlermeldung.
`$context.authorize.latency`	Die Autorisierungslatenz in ms
`$context.authorize.status`	Der Statuscode, der von einem Autorisie rungsversuch zurückgegeben wurde.
`$context.authorizer.error`	Die von einem Genehmiger zurückgegebene Fehlermeldung.

Parameter	Beschreibung
`$context.authorizer.integrationLatency`	Die Lambda-Genehmiger-Latenzzeit in ms.
`$context.authorizer.integrationStatus`	Der von einem Lambda-Genehmiger zurückgegebene Statuscode.
`$context.authorizer.latency`	Der Genehmiger-Latenz in ms.
`$context.authorizer.requestId`	Die Anforderungs-ID des AWS Endpunkts.
`$context.authorizer.status`	Der von einem Genehmiger zurückgegebene Statuscode.
`$context.authorizer.principalId`	Die ID des Prinzipalbenutzers, die mit dem vom Client gesendeten und von einer Lambda-Genehmiger-Lambda-Funktion des API Gateways zurückgegebenen Token verknüpft ist. (Ein Lambda-Genehmiger wurde früher als benutzerdefinierter Genehmiger bezeichnet).

Parameter	Beschreibung
`$context.authorizer.` *`property`*	Der in einer Zeichenfolge umgewandelte Wert des angegebenen Schlüssel-Wert-Paares der `context`-Zuordnung, der von einer API Gateway Lambda-Genehmigerfunktion zurückgegeben wird. Angenommen, der Genehmiger gibt folgende `context`-Zuweisung zurück: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` Dann gibt der Aufruf von `$context.authorizer.key` die Zeichenfolge `"value"` zurück, der Aufruf von `$context.authorizer.numKey` gibt die Zeichenfolge `"1"` zurück und der Aufruf von `$context.authorizer.boolKey` gibt den Wert `"true"` zurück.
`$context.authenticate.error`	Die von einem Authentifizierungsversuch zurückgegebene Fehlermeldung.
`$context.authenticate.latency`	Die Authentifizierungslatenz in ms
`$context.authenticate.status`	Der Statuscode, der von einem Authentifizierungsversuch zurückgegeben wurde.
`$context.connectedAt`	Die Epoch-formatierte Verbindungszeit.

Parameter	Beschreibung
`$context.connectionId`	Eine eindeutige ID für die Verbindung, die für einen Rückruf an den Client verwendet werden kann.
`$context.domainName`	Ein Domainname für die WebSocket API. Dies kann für einen Rückruf an den Client (anstelle eines hardcodierten Wertes) verwendet werden.
`$context.error.message`	Eine Zeichenfolge, die eine API Gateway-F ehlermeldung enthält.
`$context.error.messageString`	Die Wert von `$context.error.message` in Anführungszeichen, d. h. `"$context .error.message"` .
`$context.error.responseType`	Der Fehler-Antworttyp.
`$context.error.validationEr rorString`	Eine Zeichenfolge, die eine detaillierte Validieru ngsfehlermeldung enthält.
`$context.eventType`	Der Ereignistyp: `CONNECT`, `MESSAGE` oder `DISCONNECT` .
`$context.extendedRequestId`	Äquivalent mit `$context.requestId` .
`$context.identity.accountId`	Die der Anfrage zugeordnete AWS Konto-ID.
`$context.identity.apiKey`	Der API-Besitzerschlüssel, der der schlüssel fähigen API-Anforderung zugewiesen ist.
`$context.identity.apiKeyId`	Die API-Schlüssel-ID, die der schlüsselfähigen API-Anforderung zugewiesen ist

Parameter	Beschreibung
`$context.identity.caller`	Die Hauptkennung des Aufrufers, der die Anforderung signiert hat. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
`$context.identity.cognitoAuthenticationProvider`	Eine durch Komma getrennte Liste der Amazon Cognito-Authentifizierungsanbieter, die vom Aufrufer, der die Anfrage stellt, verwendet werden. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Zum Beispiel für eine Identität aus einem Amazon Cognito-Benutzerpool, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Weitere Informationen finden Sie unter Verbundidentitäten verwenden im Amazon Cognito-Entwicklerhandbuch.
`$context.identity.cognitoAuthenticationType`	Der Amazon Cognito-Authentifizierungstyp des Aufrufers, der den Anfrage erstellt hat. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde. Mögliche Werte sind `authenticated` für authentifizierte Identitäten und `unauthenticated` für nicht authentifizierte Identitäten.
`$context.identity.cognitoIdentityId`	Die Amazon Cognito Identitäts-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.

Parameter	Beschreibung
`$context.identity.cognitoId entityPoolId`	Die Amazon Cognito Identitätspool-ID des anfordernden Aufrufers. Nur verfügbar, wenn die Anfrage mit Anmeldeinformationen von Amazon Cognito signiert wurde.
`$context.identity.principalOrgId`	Die AWS -Organisations-ID. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
`$context.identity.sourceIp`	Die Quell-IP-Adresse der TCP-Verbindung, von der die Anforderung an API Gateway gesendet wird.
`$context.identity.user`	Die Hauptkennung des Benutzers, der für den Ressourcenzugriff autorisiert wird. Wird für Routen unterstützt, die die IAM-Autorisierung verwenden.
`$context.identity.userAgent`	Der Benutzeragent des API-Aufrufers.
`$context.identity.userArn`	Der ARN (Amazon Resource Name) des tatsächlichen Benutzers nach der Authentif izierung.
`$context.integration.error`	Die von einer Integration zurückgegebene Fehlermeldung.
`$context.integration.integr ationStatus`	Bei der Lambda-Proxyintegration wurde der Statuscode vom Lambda-Funktionscode zurückgegeben AWS Lambda, nicht vom Backend-Funktionscode.
`$context.integration.latency`	Die Integrationslatenz in Millisekunden. Äquivalent mit `$context.integrati onLatency` .

Parameter	Beschreibung
`$context.integration.requestId`	Die AWS Anforderungs-ID des Endpunkts . Äquivalent mit `$context.awsEndpointRequestId` .
`$context.integration.status`	Der von einer Integration zurückgegebene Statuscode. Bei Lambda-Proxy-Integrationen ist dies der Statuscode, der von Ihrem Lambda-Funktionscode zurückgegeben wird. Äquivalent mit `$context.integrationStatus` .
`$context.integrationLatency`	Die Integrationslatenz in ms, nur für die Zugriffsprotokollierung verfügbar.
`$context.messageId`	Eine eindeutige serverseitige ID für eine Nachricht. Nur verfügbar, wenn der `$context.eventType` `MESSAGE` ist.
`$context.requestId`	Entspricht `$context.extendedRequestId` .
`$context.requestTime`	Die Anforderungszeit im CLF-Format (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Die Anforderungszeit im Epoch-Format in Millisekunden.
`$context.routeKey`	Der ausgewählte Routenschlüssel.
`$context.stage`	Die Bereitstellungsstufe des API-Aufrufs (z. B. "beta" oder "prod").
`$context.status`	Der Antwortstatus.
`$context.waf.error`	Die Fehlermeldung wurde von zurückgegeben AWS WAF.
`$context.waf.latency`	Die AWS WAF Latenz in ms.

Parameter	Beschreibung
`$context.waf.status`	Der Statuscode wurde von zurückgegeben AWS WAF.

Beispiele für einige häufig verwendete Zugriffsprotokollformate werden in der API Gateway-Konsole dargestellt und wie folgt aufgeführt.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
 $context.connectionId" \
$context.status $context.requestId
```

  Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- `JSON`:

```
{
"requestId":"$context.requestId", \
"ip": "$context.identity.sourceIp", \
"caller":"$context.identity.caller", \
"user":"$context.identity.user", \
"requestTime":"$context.requestTime", \
"eventType":"$context.eventType", \
"routeKey":"$context.routeKey", \
"status":"$context.status", \
"connectionId":"$context.connectionId"
}
```

  Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- `XML`:

```
<request id="$context.requestId"> \
```

```
 <ip>$context.identity.sourceIp</ip> \
 <caller>$context.identity.caller</caller> \
 <user>$context.identity.user</user> \
 <requestTime>$context.requestTime</requestTime> \
 <eventType>$context.eventType</eventType> \
 <routeKey>$context.routeKey</routeKey> \
 <status>$context.status</status> \
 <connectionId>$context.connectionId</connectionId> \
</request>
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

- CSV (durch Komma getrennte Werte):

```
$context.identity.sourceIp,$context.identity.caller, \
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

Die Fortsetzungszeichen (\) sind als visuelle Hilfe gedacht. Das Protokollformat muss eine einzelne Zeile sein. Sie können am Ende des Protokollformats ein Zeilenumbruchzeichen (\n) hinzufügen, um am Ende jedes Protokolleintrags einen Zeilenumbruch einzuschließen.

Referenz zu API Gateway Amazon-Ressourcenname (ARN)

Die folgenden Tabellen listen die Amazon-Ressourcennamen (ARNs) für API Gateway-Ressourcen auf. Weitere Informationen zur Verwendung von ARNs in AWS Identity and Access Management Richtlinien finden Sie unter Funktionsweise von Amazon API Gateway mit IAM undKontrollieren des Zugriffs auf eine API mit IAM-Berechtigungen.

HTTP-API und WebSocket API-Ressourcen

Ressource	ARN
AccessLogSettings	`arn:partition :apigatew ay: region::/apis/api-id/ stages/stage-name /accesslo gsettings`
Api	`arn:partition :apigatew ay: region::/apis/api-id`
Apis	`arn:partition :apigatew ay: region::/apis`
ApiMapping	`arn:partition :apigatew ay: region::/domainnames/ domain-na me /apimappings/ id`
ApiMappings	`arn:partition :apigatew ay: region::/domainnames/ domain-na me /apimappings`
Authorizer	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers/ id`
Authorizers	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers`

Ressource	ARN
Cors	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/cors
Bereitstellung	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/deployme nts/ *id*
Bereitstellungen	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/deployme nts
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*
DomainNames	arn:*partition* :apigatew ay: *region*::/domainnames
ExportedAPI	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/exports/ *specification*
Integration	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ions/ *integration-id*
Integrationen	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ions
IntegrationResponse	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ionresponses/ *integration-respon se*

Ressource	ARN
IntegrationResponses	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ionresponses
Modell	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*
Modelle	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models
ModelTemplate	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*/ template
Route	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*
Routen	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes
RouteRequestParameter	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ requestparameters/ *key*
RouteResponse	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ routeresponses/ *id*
RouteResponses	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ routeresponses
RouteSettings	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name* /routeset tings/ *route-key*

Ressource	ARN
Stufe	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name*
Phasen	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/stages
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

REST-API-Ressourcen

Ressource	ARN
Account	arn:*partition* :apigatew ay: *region*::/account
ApiKey	arn:*partition* :apigatew ay: *region*::/apikeys/ *id*
ApiKeys	arn:*partition* :apigatew ay: *region*::/apikeys
Authorizer	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers/ *id*
Authorizers	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers

Ressource	ARN
BasePathMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings/ *basepath*
BasePathMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings
ClientCertificate	arn:*partition* :apigatew ay: *region*::/clientcertifica tes/ *id*
ClientCertificates	arn:*partition* :apigatew ay: *region*::/clientcertificates
Bereitstellung	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments/ *id*
Bereitstellungen	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments
DocumentationPart	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts/ *id*
DocumentationParts	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts
DocumentationVersion	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions/ *version*

Ressource	ARN
DocumentationVersions	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*
DomainNames	arn:*partition* :apigatew ay: *region*::/domainnames
GatewayResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses/ *response-type*
GatewayResponses	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses
Integration	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration
IntegrationResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration/respo nses/ *status-code*
Methode	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method*

Ressource	ARN
MethodResponse	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method* /responses/ *status-code*
Modell	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/models/*model-name*
Modelle	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/models
RequestValidator	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/requestvalidators/ *id*
RequestValidators	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/requestvalidators
Ressource	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *id*
Ressourcen	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources
RestApi	arn:*partition* :apigateway: *region*::/restapis/ *api-id*
RestApis	arn:*partition* :apigateway: *region*::/restapis

Ressource	ARN
Stufe	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages/*stage-name*
Phasen	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages
Tags	arn:*partition* :apigatew ay: *region*::/tags/*url-encoded-resource-arn*
Vorlage	arn:*partition* :apigatew ay: *region*::/restapis/models / *model-name* /template
UsagePlan	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id*
UsagePlans	arn:*partition* :apigatew ay: *region*::/usageplans
UsagePlanKey	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id* /keys/*id*
UsagePlanKeys	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id* /keys
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

execute-api(HTTP-APIs, WebSocket APIs und REST-APIs)

Ressource	ARN
WebSocket API-Endpunkt	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/`*`stage`*`/`*`route-key`*
HTTP-API- und REST-API-Endpunkt *	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/`*`stage`*`/`*`http-method`*` /`*`resource-path`*
Lambda-Genehmiger **	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/authorizers/ `*`authorizer-id`*

* Der ARN für den `$default`-Routenendpunkt für HTTP-APIs ist `arn:`*`partition`*`:execute-api:`*`region`*`:`*`account-id`*`:`*`api-id`*`/`*`*`/$default`.

** Dieser ARN ist nur für die Einrichtung der `SourceArn`-Bedingung in der Ressourcenrichtlinie für eine Lambda-Genehmiger-Funktion anwendbar. Ein Beispiel finden Sie unter the section called “Erstellen eines Lambda-Genehmigers”.

API Gateway-Erweiterungen für OpenAPI arbeiten

Die API Gateway Gateway-Erweiterungen unterstützen die AWS-spezifische Autorisierung und die API Gateway Gateway-spezifischen API-Integrationen für REST-APIs und HTTP-APIs. In diesem Abschnitt beschreiben wir die API Gateway-Erweiterungen der OpenAPI-Spezifikation.

> **Tip**
>
> Um zu verstehen, wie die API Gateway-Erweiterungen in einer Anwendung verwendet werden, können Sie die API Gateway-Konsole verwenden, um eine REST-API oder HTTP-API zu erstellen und diese in eine OpenAPI-Definitionsdatei zu exportieren. Weitere Informationen zum Exportieren einer API finden Sie unter REST-API von API Gateway importieren und Exportieren einer HTTP-API aus API Gateway.

Themen

- x-amazon-apigateway-integration.RequestParameters Objekt
- x-amazon-apigateway-integration.responses-Objekt
- x-amazon-apigateway-integration.response-Objekt
- x-amazon-apigateway-integration.responseTemplates-Objekt
- x-amazon-apigateway-integration.responseParameters-Objekt
- x-amazon-apigateway-integration.tlsConfig-Objekt
- x-amazon-apigateway-minimum-Kompressionsgröße
- x-amazon-apigateway-policy
- x-amazon-apigateway-request-validator-Eigenschaft
- x-amazon-apigateway-request-validators-Objekt
- x-amazon-apigateway-request-Validators.RequestValidator-Objekt
- x-amazon-apigateway-tag-value-Eigenschaft

x-amazon-apigateway-any-method-Objekt

Gibt das OpenAPI-Operations-Objekt für die Catch-all-Methode `ANY` von API Gateways in einem OpenAPI-Pfad-Element-Objekt an. Dieses Objekt kann neben anderen Operation-Objekten bestehen und fängt sämtliche HTTP-Methoden ab, die nicht explizit deklariert wurden.

Die folgende Tabelle listet die durch API Gateway erweiterten Eigenschaften auf. Andere OpenAPI-Operation-Eigenschaften finden Sie in der OpenAPI-Spezifikation.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`isDefaultRoute`	`Boolean`	Gibt an, ob eine Route die `$default`-Route ist. Wird nur für HTTP-APIs unterstützt. Weitere Informationen hierzu finden Sie unter Arbeiten mit Routen für HTTP-APIs.
`x-amazon-apigateway-integration`	x-amazon-apigateway-integration Objekt	Gibt die Integration der Methode mit dem Backend an. Hierbei handelt es sich

Name der Eigenschaft	Typ	Beschreibung
		um eine erweiterte Eigenschaft des Objekts OpenAPI-Operation. Die Integration kann vom Typ `AWS`, `AWS_PROXY`, `HTTP`, `HTTP_PROXY` oder `MOCK` sein.

x-amazon-apigateway-any-Methodenbeispiele

Im folgenden Beispiel wird die `ANY`-Methode auf einer Proxy-Ressource `{proxy+}` mit einer Lambda-Funktion `TestSimpleProxy` integriert.

```
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:TestSimpleProxy/invocations",
          "httpMethod": "POST",
          "type": "aws_proxy"
        }
```

Das folgende Beispiel erstellt eine `$default`-Route für eine HTTP-API, die sich in eine Lambda-Funktion (`HelloWorld`) integriert.

```
"/$default": {
    "x-amazon-apigateway-any-method": {
```

```
      "isDefaultRoute": true,
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
        "timeoutInMillis": 1000,
        "connectionType": "INTERNET",
        "payloadFormatVersion": 1.0
      }
   }
}
```

x-amazon-apigateway-cors Objekt

Gibt die CORS-Konfiguration (Cross-Origin Resource Sharing) für eine HTTP-API an. Die Erweiterung gilt für die OpenAPI-Struktur auf Stammebene. Weitere Informationen hierzu finden Sie unter Konfigurieren von CORS für eine HTTP-API.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`allowOrigins`	`Array`	Gibt die zulässigen Ursprünge an.
`allowCredentials`	`Boolean`	Gibt an, ob Anmeldeinformationen in der CORS-Anforderung enthalten sind.
`exposeHeaders`	`Array`	Gibt die Header an, die bereitgestellt werden.
`maxAge`	`Integer`	Gibt die Anzahl der Sekunden an, während der der Browser Preflight-Anfrageergebnisse zwischenspeichern soll.
`allowMethods`	`Array`	Gibt die zulässigen HTTP-Methoden an.

Name der Eigenschaft	Typ	Beschreibung
`allowHeaders`	`Array`	Gibt die zulässigen Header an.

x-amazon-apigateway-cors Beispiel

Das folgende Beispiel zeigt eine CORS-Konfiguration für eine HTTP-API.

```
"x-amazon-apigateway-cors": {
    "allowOrigins": [
      "https://www.example.com"
    ],
    "allowCredentials": true,
    "exposeHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ],
    "maxAge": 3600,
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ]
}
```

x-amazon-apigateway-apiEigenschaft -key-source

Geben Sie die Quelle eines API-Schlüssels an, um die API-Methoden, die einen Schlüssel fordern, zu drosseln. Diese Eigenschaft auf API-Ebene ist vom Typ `String`. Weitere Hinweise zur Konfiguration einer Methode, für die ein API-Schlüssel erforderlich ist, finden Sie unter. the section called “Konfigurieren Sie eine Methode zur Verwendung von API-Schlüsseln mit einer OpenAPI-Definition”

Geben Sie die Quelle des API-Schlüssels für Anforderungen an. Folgende Werte sind zulässig:

- `HEADER` für den Empfang des API-Schlüssels aus dem `X-API-Key`-Header einer Anforderung.
- `AUTHORIZER` für den Empfang des API-Schlüssels von `UsageIdentifierKey` von einem Lambda Genehmiger (ehemals als benutzerdefinierter Genehmiger bezeichnet).

x-amazon-apigateway-apiBeispiel für -key-source

Im folgenden Beispiel wird der `X-API-Key`-Header als Quelle des API-Schlüssels festgelegt.

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "title" : "Test1"
   },
  "schemes" : [ "https" ],
  "basePath" : "/import",
  "x-amazon-apigateway-api-key-source" : "HEADER",
   .
   .
   .
}
```

OpenAPI 3.0.1

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Test1"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "import"
      }
    }
  } ],
  "x-amazon-apigateway-api-key-source" : "HEADER",
```

```
    .
    .
    .
}
```

x-amazon-apigateway-auth Objekt

Definiert einen Autorisierungstyp, der für die Autorisierung von Methodenaufrufen in API Gateway angewendet werden soll.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`type`	`string`	Gibt den Autorisierungstyp an. Geben Sie `"NONE"` für einen offenen Zugriff an. Geben Sie `"AWS_IAM"` an, um IAM-Berechtigungen zu verwenden . Bei Werten wird die Groß- und Kleinschreibung nicht berücksichtigt.

x-amazon-apigateway-auth Beispiel

Das folgende Beispiel legt den Autorisierungstyp für eine API-Methode fest.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "openapi3",
    "version": "1.0"
  },
  "paths": {
    "/protected-by-iam": {
      "get": {
        "x-amazon-apigateway-auth": {
```

```
          "type": "AWS_IAM"
        }
      }
    }
  }
}
```

x-amazon-apigateway-authorizer Objekt

Definiert einen Lambda-Genehmiger, Amazon Cognito-Benutzerpool oder JWT-Genehmiger, der für die Autorisierung von Methodenaufrufen in API Gateway angewendet werden soll. Diese Erweiterung gilt für die Sicherheitsdefinition in OpenAPI 2 und OpenAPI 3.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`type`	`string`	Der Typ des Genehmigers. Diese Eigenschaft ist erforderlich. Für REST-APIs geben Sie `token` für einen Genehmiger an, bei dem die Identität des Aufrufers in ein Autorisierungs-Token eingebettet ist. Geben Sie `request` für einen Genehmiger an, wobei die Aufruferidentität in den Anforderungsparametern enthalten ist. Geben Sie `cognito_user_pools` für einen Genehmiger an, der einen Amazon Cognito-Benutzerpool verwendet, um den Zugriff auf Ihre API zu kontrollieren.

Name der Eigenschaft	Typ	Beschreibung
		Für HTTP-APIs geben Sie `request` für einen Lambda-Genehmiger mit der in Anfrageparametern enthalten en Aufruferidentität an. Geben Sie `jwt` für einen JWT-Geneh miger an.
`authorizerUri`	`string`	Der Uniform Resource Identifier (URI) der Lambda-Fu nktion des Genehmigers. Die Syntax ist wie folgt: `"arn:aws:apigatewa y:us-east-1:lambda :path/2015-03-31/ functions/ arn:aws:lambda:us- east-1: account-i d :function: auth_func tion_name /invocati ons"`
`authorizerCredenti als`	`string`	Für den Aufruf des Genehmigers erforderl iche Anmeldeinformation en, sofern vorhanden, in Form eines ARN einer IAM-Ausführungsrolle. Beispiel: "arn:aws:iam::*account-i d* :*IAM_role*".

Name der Eigenschaft	Typ	Beschreibung
`authorizerPayloadFormatVersion`	`string`	Für HTTP-APIs geben Sie das Format der Daten an, die API Gateway an einen Lambda-Genehmiger sendet, und wie API Gateway die Antwort von Lambda interpretiert. Weitere Informationen hierzu finden Sie unter the section called “Nutzlastformatversion”.
`enableSimpleResponses`	`Boolean`	Für HTTP-APIs geben Sie an, ob ein `request`-Genehmiger einen booleschen Wert oder eine IAM-Richtlinie zurückgibt. Wird nur für Genehmiger mit `authorizerPayloadFormatVersion 2.0` unterstützt. Falls aktiviert, gibt die Lambda-Genehmiger-Funktion einen booleschen Wert zurück. Weitere Informationen hierzu finden Sie unter the section called “Lambda-Funktionsantwort für Format 2.0”.
`identitySource`	`string`	Eine CSV-Liste der Mapping-Ausdrücke der Anforderungsparameter als Identitätsquelle. Gilt nur für die Genehmiger des Typs `request` und `jwt`.

Name der Eigenschaft	Typ	Beschreibung
`jwtConfiguration`	`Object`	Gibt den Aussteller und die Zielgruppen für einen JWT-Genehmiger an. Weitere Informationen finden Sie unter JWTConfiguration in der API Gateway Version 2 API-Refer enz. Wird nur für HTTP-APIs unterstützt.
`identityValidation Expression`	`string`	Ein regulärer Ausdruck für die Validierung des Tokens als eingehende Identität. Beispiel: "^x-[a-z]+". Wird nur für `TOKEN` Autorisierer für REST-APIs unterstützt.
`authorizerResultTt lInSeconds`	`string`	Die Anzahl der Sekunden, über die hinweg das Genehmiger-Ergebnis zwischengespeichert wird.
`providerARNs`	Ein Array von `string`	Eine Liste der Amazon-Co gnito-Benutzerpool-ARNs für den `COGNITO_USER_POOLS` .

x-amazon-apigateway-authorizer Beispiele für REST-APIs

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen gibt einen Lambda-Genehmiger des Typs "token" des Namens a `test-authorizer`.

```
  "securityDefinitions" : {
    "test-authorizer" : {
      "type" : "apiKey",                          // Required and the value must be
 "apiKey" for an API Gateway API.
```

```
      "name" : "Authorization",                  // The name of the header containing
 the authorization token.
      "in" : "header",                           // Required and the value must be
 "header" for an API Gateway API.
      "x-amazon-apigateway-authtype" : "oauth2", // Specifies the authorization
 mechanism for the client.
      "x-amazon-apigateway-authorizer" : {       // An API Gateway Lambda authorizer
 definition
        "type" : "token",                        // Required property and the value
 must "token"
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
        "authorizerCredentials" : "arn:aws:iam::account-id:role",
        "identityValidationExpression" : "^x-[a-z]+",
        "authorizerResultTtlInSeconds" : 60
      }
    }
  }
```

Das folgende OpenAPI-Operations-Objekt-Snippet legt `GET /http` auf die Verwendung des vorhergehenden Lambda-Genehmigers fest.

```
  "/http" : {
    "get" : {
      "responses" : { },
      "security" : [ {
        "test-authorizer" : [ ]
      } ],
      "x-amazon-apigateway-integration" : {
        "type" : "http",
        "responses" : {
          "default" : {
            "statusCode" : "200"
          }
        },
        "httpMethod" : "GET",
        "uri" : "http://api.example.com"
      }
    }
  }
```

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen spezifiziert einen Lambda-Genehmiger des Typs "request" mit einem einzigen Header-Parameter (`auth`) als Identitätsquelle. Die `securityDefinitions` heißt `request_authorizer_single_header`.

```
"securityDefinitions": {
    "request_authorizer_single_header" : {
      "type" : "apiKey",
      "name" : "auth",              // The name of a single header or query parameter
 as the identity source.
      "in" : "header",              // The location of the single identity source
 request parameter. The valid value is "header" or "query"
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "method.request.header.auth",   // Request parameter mapping
 expression of the identity source. In this example, it is the 'auth' header.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
}
```

Das folgende Beispiel für OpenAPI-Sicherheitsdefinitionen spezifiziert einen Lambda-Genehmiger des Typs "request" mit einem Header (`HeaderAuth1`) und einem Abfragezeichenfolgenparameter `QueryString1` als Identitätsquellen.

```
"securityDefinitions": {
    "request_authorizer_header_query" : {
      "type" : "apiKey",
      "name" : "Unused",            // Must be "Unused" for multiple identity sources
 or non header or query type of request parameters.
      "in" : "header",              // Must be "header" for multiple identity sources
 or non header or query type of request parameters.
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
```

```
        "identitySource" : "method.request.header.HeaderAuth1,
 method.request.querystring.QueryString1",   // Request parameter mapping expressions
 of the identity sources.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
}
```

Das folgende OpenAPI-Sicherheitsdefinitionsbeispiel gibt einen API Gateway Lambda-Genehmiger vom Typ "request" mit einer einstufigen Variablen (`stage`) als Identitätsquelle an.

```
"securityDefinitions": {
    "request_authorizer_single_stagevar" : {
      "type" : "apiKey",
      "name" : "Unused",             // Must be "Unused", for multiple identity sources
 or non header or query type of request parameters.
      "in" : "header",               // Must be "header", for multiple identity sources
 or non header or query type of request parameters.
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "stageVariables.stage",   // Request parameter mapping
 expression of the identity source. In this example, it is the stage variable.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
}
```

Das folgende Beispiel für eine OpenAPI-Sicherheitsdefinition gibt einen Amazon Cognito-Benutzerpool als Genehmiger an.

```
 "securityDefinitions": {
    "cognito-pool": {
```

```
      "type": "apiKey",
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "cognito_user_pools",
      "x-amazon-apigateway-authorizer": {
        "type": "cognito_user_pools",
        "providerARNs": [
          "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-east-1_ABC123"
        ]
      }
    }
```

Der folgende OpenAPI-Operationsobjektausschnitt legt `GET /http` fest, um den vorherigen Amazon Cognito-Benutzerpool als Genehmiger ohne benutzerdefinierte Bereiche zu verwenden.

```
  "/http" : {
     "get" : {
       "responses" : { },
       "security" : [ {
         "cognito-pool" : [ ]
       } ],
       "x-amazon-apigateway-integration" : {
         "type" : "http",
         "responses" : {
           "default" : {
             "statusCode" : "200"
           }
         },
         "httpMethod" : "GET",
         "uri" : "http://api.example.com"
       }
     }
    }
```

x-amazon-apigateway-authorizer Beispiele für HTTP-APIs

Das folgende OpenAPI 3.0-Beispiel erstellt einen JWT-Genehmiger für eine HTTP-API, die Amazon Cognito als Identitätsanbieter verwendet, mit dem `Authorization`-Header als Identitätsquelle.

```
"securitySchemes": {
  "jwt-authorizer-oauth": {
```

```
    "type": "oauth2",
     "x-amazon-apigateway-authorizer": {
       "type": "jwt",
       "jwtConfiguration": {
          "issuer": "https://cognito-idp.region.amazonaws.com/userPoolId",
          "audience": [
            "audience1",
            "audience2"
          ]
        },
        "identitySource": "$request.header.Authorization"
    }
  }
}
```

Im folgenden Beispiel für OpenAPI 3.0 wird derselbe JWT-Genehmiger erstellt wie im vorherigen Beispiel. In diesem Beispiel wird jedoch die OpenAPI-`openIdConnectUrl`-Eigenschaft verwendet, damit der Aussteller automatisch erkannt wird. Die `openIdConnectUrl` muss vollständig gebildet sein.

```
"securitySchemes": {
  "jwt-authorizer-autofind": {
    "type": "openIdConnect",
    "openIdConnectUrl": "https://cognito-idp.region.amazonaws.com/userPoolId/.well-
known/openid-configuration",
    "x-amazon-apigateway-authorizer": {
      "type": "jwt",
      "jwtConfiguration": {
        "audience": [
          "audience1",
          "audience2"
        ]
      },
      "identitySource": "$request.header.Authorization"
    }
  }
}
```

Das folgende Beispiel erstellt einen Lambda-Genehmiger für eine HTTP-API. Dieser Beispiel-Genehmiger verwendet den `Authorization`-Header als Identitätsquelle. Der Genehmiger verwendet Nutzlast-Formatversion `2.0` und gibt den booleschen Wert zurück, da `enableSimpleResponses` auf `true` gesetzt ist.

```
"securitySchemes" : {
  "lambda-authorizer" : {
    "type" : "apiKey",
    "name" : "Authorization",
    "in" : "header",
    "x-amazon-apigateway-authorizer" : {
      "type" : "request",
      "identitySource" : "$request.header.Authorization",
      "authorizerUri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:function-name/invocations",
      "authorizerPayloadFormatVersion" : "2.0",
      "authorizerResultTtlInSeconds" : 300,
      "enableSimpleResponses" : true
    }
  }
}
```

x-amazon-apigateway-authtype Eigentum

Bei REST-APIs kann diese Erweiterung verwendet werden, um einen benutzerdefinierten Typ eines Lambda-Genehmigers zu definieren. In diesem Fall liegt der Wert in Freiform vor. Beispielsweise kann eine API mehrere Lambda-Genehmiger haben, die verschiedene interne Schemata verwenden. Sie können diese Erweiterung verwenden, um das interne Schema eines Lambda-Genehmigers zu identifizieren.

In HTTP-APIs und REST-APIs kann es häufiger auch verwendet werden, um die IAM-Genehmigung für mehrere Vorgänge zu definieren, die dasselbe Sicherheitsschema verwenden. In diesem Fall ist der Begriff `awsSigv4` ein reservierter Begriff, zusammen mit jedem Begriff, dem ein Präfix `aws` vorangestellt ist.

Diese Erweiterung gilt für das Sicherheitsschema des Typs `apiKey` in OpenAPI 2 und OpenAPI 3.

x-amazon-apigateway-authtype Beispiel

Das folgende OpenAPI 3-Beispiel definiert die IAM-Genehmigung für mehrere Ressourcen in einer REST-API oder HTTP API:

```
{
  "openapi" : "3.0.1",
  "info" : {
```

```
    "title" : "openapi3",
    "version" : "1.0"
  },
  "paths" : {
    "/operation1" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    },
    "/operation2" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "sigv4Reference" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authtype": "awsSigv4"
      }
    }
  }
}
```

Das folgende OpenAPI 3-Beispiel definiert einen Lambda-Genehmiger mit einem benutzerdefinierten Schema für eine REST-API:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3 for REST API",
    "version" : "1.0"
  },
  "paths" : {
    "/protected-by-lambda-authorizer" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "myAuthorizer" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "myAuthorizer" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authorizer" : {
          "identitySource" : "method.request.header.Authorization",
          "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
          "authorizerResultTtlInSeconds" : 300,
          "type" : "request",
          "enableSimpleResponses" : false
        },
        "x-amazon-apigateway-authtype": "Custom scheme with corporate claims"
      }
    }
  },
  "x-amazon-apigateway-importexport-version" : "1.0"
}
```

Weitere Informationen finden Sie auch unter

authorizer.authType

x-amazon-apigateway-binaryEigenschaft -media-types

Gibt die Liste der binären Medientypen an, die von API Gateway unterstützt werden sollen, z. B. `application/octet-stream` und `image/jpeg`. Diese Erweiterung ist ein JSON-Array. Dies sollte als Top-Level-Erweiterung des Herstellers im OpenAPI-Dokument eingeschlossen werden.

x-amazon-apigateway-binaryBeispiel für -Medientypen

Das folgende Beispiel zeigt die Codierungs-Suchreihenfolge einer API.

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

x-amazon-apigateway-documentation Objekt

Definiert die Dokumentationsteile, die in API Gateway importiert werden sollen. Dieses Objekt ist ein JSON-Objekt, das ein Array der `DocumentationPart`-Instances enthält.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`documentationParts`	`Array`	Ein Array der exportierten oder importierten `DocumentationPart` -Instances.
`version`	`String`	Die Versions-ID des Snapshots der exportierten Dokumentationsbausteine.

x-amazon-apigateway-documentation Beispiel

Das folgende Beispiel einer API Gateway-Erweiterung für OpenAPI definiert `DocumentationParts`-Instances, die in eine API in API Gateway importiert oder von einer API in API Gateway exportiert werden sollen.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
       {
         "location": {
          "type": "API"
       },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
         … // Another DocumentationPart instance
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-Konfigurationsobjekt

Gibt Details der Endpunktkonfiguration für eine API an. Diese Erweiterung ist eine erweiterte Eigenschaft des Objekts OpenAPI-Operation. Dieses Objekt sollte in Top-Level-Anbietererweiterungen für Swagger 2.0 vorhanden sein. Für OpenAPI 3.0 sollte es unter den Anbietererweiterungen des Server-Objekts vorhanden sein.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`disableExecuteApiEndpoint`	Boolesch	Gibt an, ob Clients Ihre API mithilfe des `execute-api` -Standardendpunkts aufrufen können. Standardmäßig können Kunden Ihre API mit dem standardmäßigen

Name der Eigenschaft	Typ	Beschreibung
		`https://{api_id}.execute-api.{region}.amazonaws.com` - Endpunkt aufrufen. Wenn Sie erzwingen möchten, dass Kunden einen benutzerdefinierten Domänennamen verwenden, um Ihre API aufzurufen, geben Sie a `true`.
`vpcEndpointIds`	Ein Array von `String`	Eine Liste von VpcEndpoint Bezeichnern, anhand derer Route 53-Aliaseinträge für eine REST-API erstellt werden sollen. Wird nur für REST-APIs vom Endpunkttyp `PRIVATE` unterstützt.

x-amazon-apigateway-endpoint-Konfigurationsbeispiele

Im folgenden Beispiel werden die angegebenen VPC-Endpunkte der REST-API zugeordnet.

```
"x-amazon-apigateway-endpoint-configuration": {
    "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

Das folgende Beispiel deaktiviert den Standardendpunkt für eine API.

```
"x-amazon-apigateway-endpoint-configuration": {
    "disableExecuteApiEndpoint": true
}
```

x-amazon-apigateway-gatewayObjekt -Responses

Definiert die Gateway-Antworten für eine API als eine String-zu-Zuordnung von GatewayResponseSchlüssel-Wert-Paaren. Die Erweiterung gilt für die OpenAPI-Struktur auf Stammebene.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
responseType	x-amazon-apigateway-gateway-responses.GatewayResponse	Eine GatewayResponse für den angegebenen *responseType* .

x-amazon-apigateway-gatewayBeispiel für -Antworten

Die folgende API Gateway Gateway-Erweiterung für das OpenAPI-Beispiel definiert eine GatewayResponsesMap, die zwei GatewayResponseInstanzen enthält — eine für den DEFAULT_4XX Typ und eine weitere für den Typ. INVALID_API_KEY

```
{
  "x-amazon-apigateway-gateway-responses": {
    "DEFAULT_4XX": {
      "responseParameters": {
        "gatewayresponse.header.Access-Control-Allow-Origin": "'domain.com'"
      },
      "responseTemplates": {
        "application/json": "{\"message\": test 4xx b }"
      }
    },
    "INVALID_API_KEY": {
      "statusCode": "429",
      "responseTemplates": {
        "application/json": "{\"message\": test forbidden }"
      }
    }
  }
}
```

x-amazon-apigateway-gateway-Responses.GatewayResponse-Objekt

Definiert eine Gateway-Antwort eines bestimmten Antworttyps, einschließlich des Statuscodes, sämtlicher geltender Antwortparameter oder Antwortvorlagen.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
responseParameters	x-amazon-apigateway-gateway-Responses.Response-Parameter	Spezifiziert die Parameter, nämlich die GatewayResponseHeader-Parameter. Die Parameterwerte können jeden beliebigen eingehenden request parameter-Wert oder einen statischen benutzerdefinierten Wert übernehmen.
responseTemplates	x-amazon-apigateway-gateway-Responses.ResponseTemplates	Gibt die Mapping-Vorlagen der Gateway-Antwort an. Die Vorlagen werden nicht von der VTL-Engine verarbeitet.
statusCode	`string`	Ein HTTP-Statuscode für die Gateway-Antwort.

x-amazon-apigateway-gateway-Responses.GatewayResponse-Beispiel

Das folgende Beispiel für die API Gateway Gateway-Erweiterung für OpenAPI definiert, GatewayResponsedass die `INVALID_API_KEY` Antwort so angepasst werden soll, dass sie den Statuscode von`456`, den `api-key` Header-Wert der eingehenden Anfrage und eine `"Bad api-key"` Nachricht zurückgibt.

```
"INVALID_API_KEY": {
  "statusCode": "456",
  "responseParameters": {
```

```
        "gatewayresponse.header.api-key": "method.request.header.api-key"
      },
      "responseTemplates": {
        "application/json": "{\"message\": \"Bad api-key\" }"
      }
    }
```

x-amazon-apigateway-gateway-Responses.ResponseParameters-Objekt

Definiert eine string-to-string Zuordnung von Schlüssel-Wert-Paaren, um Gateway-Antwortparameter aus den eingehenden Anforderungsparametern oder unter Verwendung von Literalzeichenfolgen zu generieren. Wird nur für REST-APIs unterstützt.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`gatewayresponse.` *`param-position`* `.` *`param-name`*	`string`	*`param-position`* kann `header`, `path` oder `querystring` sein. Weitere Informationen finden Sie unter Zuweisen von Methodenanforderungsdaten zu Integrationsanforderungs-Parametern.

x-amazon-apigateway-gateway-Responses.ResponseParameters — Beispiel

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt einen Ausdruck für die Zuordnung von GatewayResponseAntwortparametern, um die CORS-Unterstützung für Ressourcen auf den `*.example.domain` Domänen zu aktivieren.

```
    "responseParameters": {
      "gatewayresponse.header.Access-Control-Allow-Origin": '*.example.domain',
      "gatewayresponse.header.from-request-header" : method.request.header.Accept,
      "gatewayresponse.header.from-request-path" : method.request.path.petId,
```

```
        "gatewayresponse.header.from-request-query" : method.request.querystring.qname
      }
```

x-amazon-apigateway-gateway-Responses.ResponseTemplates-Objekt

Definiert GatewayResponseZuordnungsvorlagen als string-to-string Zuordnung von Schlüssel-Wert-Paaren für eine bestimmte Gateway-Antwort. Für jedes Schlüssel-Wert-Paar ist der Schlüssel der Inhaltstyp. Zum Beispiel ist "application/json" und der Wert eine "stringify"-Mapping-Vorlage für einfache Variablenersetzungen. Eine `GatewayResponse`-Mapping-Vorlage wird nicht vom Velocity Template Language (VTL)-Modul verarbeitet.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`content-type`	`string`	Eine `GatewayResponse` - Text-Mapping-Vorlage, die nur einfache Variablen ersetzungen unterstützt, um einen Gateway-Antworttext anzupassen.

x-amazon-apigateway-gateway-Responses.ResponseTemplates — Beispiel

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt eine GatewayResponseMapping-Vorlage zum Anpassen einer vom API Gateway generierten Fehlerantwort in ein anwendungsspezifisches Format.

```
      "responseTemplates": {
        "application/json": "{ \"message\": $context.error.messageString, \"type\":
$context.error.responseType, \"statusCode\": '488' }"
      }
```

Das folgende Beispiel für OpenAPI-Erweiterungen zeigt eine GatewayResponseMapping-Vorlage zum Überschreiben einer vom API Gateway generierten Fehlerantwort mit einer statischen Fehlermeldung.

```
    "responseTemplates": {
      "application/json": "{ \"message\": 'API-specific errors' }"
    }
```

x-amazon-apigateway-importexport-Version

Gibt die Version des Import- und Export-Algorithmus von API Gateway für HTTP-APIs an. Derzeit wird als einziger Wert unterstütz `1.0`. Weitere Informationen finden Sie unter exportVersion in der API Gateway Version 2 API-Referenz.

x-amazon-apigateway-importexportBeispiel für eine Version

Im folgenden Beispiel wird die Import- und Exportversion auf festgeleg `1.0`.

```
{
    "openapi": "3.0.1",
    "x-amazon-apigateway-importexport-version": "1.0",
    "info": { ...
```

x-amazon-apigateway-integration Objekt

Gibt die Details der für diese Methode verwendeten Backend-Integration an. Diese Erweiterung ist eine erweiterte Eigenschaft des Objekts OpenAPI-Operation. Das Ergebnis ist ein API Gateway-Integration-Objekt.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`cacheKeyParameters`	Ein Array von `string`	Eine Liste an Anforderungsparametern, deren Werte zwischengespeichert werden sollen.
`cacheNamespace`	`string`	Eine API-spezifische Tag-Gruppe zugehöriger

Name der Eigenschaft	Typ	Beschreibung
		zwischengespeicherter Parameter.
`connectionId`	`string`	Die ID von a VpcLinkfür die private Integration.
`connectionType`	`string`	Der Verbindungstyp der Integration. Der gültige Wert ist `"VPC_LINK"` für die private Integration oder andernfalls `"INTERNET"` .

Name der Eigenschaft	Typ	Beschreibung
`credentials`	`string`	Geben Sie für rollenbasierte AWS IAM-Anmeldeinformationen den ARN einer entsprechenden IAM-Rolle an. Wenn sie nicht angegeben sind, werden als Anmeldeinformationen standardmäßig ressourcenbasierte Berechtigungen festgelegt, die manuell hinzugefügt werden, damit die API auf die Ressource zugreifen kann. Weitere Informationen finden Sie unter Gewähren von Berechtigungen mit einer Ressourcenrichtlinie. Hinweis: Stellen Sie bei der Verwendung von IAM-Anmeldeinformationen sicher, dass regionale AWS -STS-Endpunkte für die Region aktiviert sind, in der diese API für maximale Leistung bereitgestellt wird.

Name der Eigenschaft	Typ	Beschreibung
`contentHandling`	`string`	Fordern Sie Nutzlast-Kodierungs-Umwandlungstypen an. Gültige Werte sind 1) `CONVERT_TO_TEXT` für die Umwandlung einer binären Nutzlast in eine base64-kodierte Zeichenfolge oder die Umwandlung einer Textnutzlast in eine `utf-8`-kodierte Zeichenfolge oder die native Weiterleitung der Textnutzlast ohne Änderung und 2) `CONVERT_TO_BINARY` für die Umwandlung einer Textnutzlast in einen base64-dekodierten Blob oder die native Weiterleitung einer binären Nutzlast ohne Änderung.
`httpMethod`	`string`	Die in der Integrationsanforderung verwendete HTTP-Methode. Für Lambda-Funktionsaufrufe muss der Wert sei `POST`.
`integrationSubtype`	`string`	Gibt den Integrationsuntertyp für eine Serviceintegration an. AWS Wird nur für HTTP-APIs unterstützt. Unterstützte Integration-Subtypen finden Sie unter the section called “AWS Referenz zu Serviceintegrationen”.

Name der Eigenschaft	Typ	Beschreibung
`passthroughBehavior`	`string`	Gibt an, wie eine Anforderungsnutzlast eines nicht zugeordneten Inhaltstyps unverändert an die Integrationsanforderung weitergeleitet wurde. Unterstützte Werte sind `when_no_templates`, `when_no_match` und `never`. Weitere Informationen finden Sie unter Integration.passthroughBehavior.
`payloadFormatVersion`	`string`	Gibt das Format der an eine Integration gesendeten Nutzlast an. Erforderlich für HTTP-APIs. Für HTTP-APIs lauten unterstützte Werte für Lambda-Proxy-Integrationen `1.0` und `2.0`. Für alle anderen Integrationen ist `1.0` der einzige unterstützte Wert. Weitere Informationen hierzu finden Sie unter the section called “AWS Lambda Integrationen” und the section called “AWS Referenz zu Serviceintegrationen”.

Name der Eigenschaft	Typ	Beschreibung
`requestParameters`	x-amazon-apigateway-integration.RequestParameters Objekt	Gibt für REST-APIs Mappings von Methodenanforderungsparametern zu Integrationsanforderungsparametern an. Die unterstützten Anforderungsparameter sind `querystring` , `path`, `header` und `body`. Bei HTTP-APIs sind Anforderungsparameter eine Schlüssel-Wert-Zuordnung, die Parameter angibt, die an `AWS_PROXY` -Integrationen mit einem angegebenen `integrationSubtype` übergeben werden. Sie können statische Werte angeben oder Anforderungsdaten, Stufenvariablen oder Kontextvariablen zuordnen, die zur Laufzeit ausgewertet werden. Weitere Informationen hierzu finden Sie unter the section called “AWS Serviceintegrationen”.
`requestTemplates`	x-amazon-apigateway-integration.requestTemplates-Objekt	Mapping-Vorlagen für eine Anforderungsnutzlast angegebener MIME-Typen.

Name der Eigenschaft	Typ	Beschreibung
`responses`	x-amazon-apigateway-integration.responses-Objekt	Definiert die Antworten der Methode und gibt gewünschte Parameter-Mappings oder Nutzlast-Mappings von Integrationsantworten auf Methodenantworten an.
`timeoutInMillis`	`integer`	Integrationszeitüberschreitungen zwischen 50 ms und 29.000 ms.

Name der Eigenschaft	Typ	Beschreibung
`type`	`string`	Der Typ der Integration mit dem angegebenen Backend. Folgende Werte sind zulässig: • `http` oder `http_proxy` , für die Integration mit einem HTTP-Backend. • `aws_proxy` , für die Integration mit AWS Lambda-Funktionen. • `aws`, für die Integration mit AWS Lambda-Fu nktionen oder anderen AWS Diensten wie Amazon DynamoDB, Amazon Simple Notification Service oder Amazon Simple Queue Service. • `mock` für die Integration mit API Gateway, ohne ein Backend aufzurufen. Weitere Informationen über die Integrationstypen finden Sie unter integration:type.
`tlsConfig`	the section called “x-amazon -apigateway-integration.tls Config”	Gibt die TLS-Konfiguration für eine Integration an.

Name der Eigenschaft	Typ	Beschreibung
`uri`	`string`	Die Endpunkt-URI des Backends. Für Integrationen des `aws`-Typs ist es ein ARN-Wert. Für die HTTP-Integration ist dies die URL des HTTP-Endpunkts einschließlich des `https`- oder `http`-Schemas.

x-amazon-apigateway-integration Beispiele

Für HTTP-APIs können Sie Integrationen im Komponenten-Abschnitt Ihrer OpenAPI-Definition definieren. Weitere Informationen hierzu finden Sie unter x-amazon-apigateway-integrations Objekt.

```
"x-amazon-apigateway-integration": {
    "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

Im folgenden Beispiel wird eine Integration mit einer Lambda-Funktion erstellt. Zu Demonstrationszwecken wird für die Beispiel-Mapping-Vorlagen, die in `requestTemplates` und `responseTemplates` der Beispiele unten gezeigt werden, angenommen, dass die folgenden JSON-formatierten Nutzlasten angewendet werden: `{ "name":"value_1", "key":"value_2", "redirect": {"url" :"..."} }` zur Generierung einer JSON-Ausgabe von `{ "stage":"value_1", "user-id":"value_2" }` oder einer XML-Ausgabe von `<stage>value_1</stage>`.

```
"x-amazon-apigateway-integration" : {
    "type" : "aws",
    "uri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:012345678901:function:HelloWorld/invocations",
    "httpMethod" : "POST",
    "credentials" : "arn:aws:iam::012345678901:role/apigateway-invoke-lambda-exec-
role",
    "requestTemplates" : {
```

```
                "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
                "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
    },
    "requestParameters" : {
        "integration.request.path.stage" : "method.request.querystring.version",
        "integration.request.querystring.provider" :
 "method.request.querystring.vendor"
    },
    "cacheNamespace" : "cache namespace",
    "cacheKeyParameters" : [],
    "responses" : {
        "2\\d{2}" : {
            "statusCode" : "200",
            "responseParameters" : {
                "method.response.header.requestId" : "integration.response.header.cid"
            },
            "responseTemplates" : {
                "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
                "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
            }
        },
        "302" : {
            "statusCode" : "302",
            "responseParameters" : {
                "method.response.header.Location" :
 "integration.response.body.redirect.url"
            }
        },
        "default" : {
            "statusCode" : "400",
            "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static value'"
            }
        }
    }
}
```

Beachten Sie, dass doppelte Anführungszeichen (") für die JSON-Zeichenfolge in den Mapping-Vorlagen durch einen umgekehrten Schrägstrich (\") geschützt sein müssen.

x-amazon-apigateway-integrations Objekt

Definiert eine Sammlung von Integrationen. Sie können Integrationen im Komponentenbereich Ihrer OpenAPI-Definition definieren und die Integrationen für mehrere Routen wiederverwenden. Wird nur für HTTP-APIs unterstützt.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`Integration`	x-amazon-apigateway-integration Objekt	Eine Sammlung von Integrationsobjekten.

x-amazon-apigateway-integrations Beispiel

Im folgenden Beispiel wird eine HTTP-API erstellt, die zwei Integrationen definiert und die Integrationen über referenzier `$ref": "#/components/x-amazon-apigateway-integrations/`*`integration-name`*.

```
{
  "openapi": "3.0.1",
  "info":
    {
      "title": "Integrations",
      "description": "An API that reuses integrations",
      "version": "1.0"
    },
  "servers": [
  {
    "url": "https://example.com/{basePath}",
    "description": "The production API server",
    "variables":
      {
        "basePath":
          {
            "default": "example/path"
          }
```

```
      }
  }],
  "paths":
    {
      "/":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/pets":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/checkout":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration2"
                }
            }
        }
    },
    "components": {
      "x-amazon-apigateway-integrations":
        {
          "integration1":
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
```

```
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion": "1.0"
            },
          "integration2":
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:example-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion" : "1.0"
            }
        }
    }
}
```

x-amazon-apigateway-integration.requestTemplates-Objekt

Gibt die Mapping-Vorlagen für eine Anforderungsnutzlast der angegebenen MIME-Typen an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
MIME type	`string`	Ein Beispiel für den MIME-Typ ist `application/json`. Weitere Informationen zum Erstellen einer Mapping-Vorlage finden Sie unter PetStore Vorlage für eine Zuordnung.

x-amazon-apigateway-integration.requestTemplates Beispiel

Im folgenden Beispiel werden Mapping-Vorlagen für eine Anforderungsnutzlast der `application/json`- und `application/xml`-MIME-Typen festgelegt.

```
"requestTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration.RequestParameters Objekt

Für REST-APIs werden Mappings von benannten Methodenanforderungsparametern zu Integrationsanforderungsparametern angegeben. Die Methodenanforderungsparameter müssen definiert werden, bevor auf sie verwiesen wird.

Für HTTP-APIs werden Parameter angegeben, die mit einem angegebenen `AWS_PROXY` an `integrationSubtype`-Integrationen übergeben werden.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`integration.request.` *<param-type>* `.` *<param-name>*	`string`	Bei REST-APIs ist der Wert normalerweise ein vordefinierter Methodenanforderungsparameter im `method.request.` *<param-type>* `.` *<param-name>* -Format, wobei `<param-type>` `querystring` , `path`, `header` oder `body` sein kann. `$context.` *VARIABLE_NAME* , `$stageVariables.` *VARIABLE_NAME* und *STATIC_VALUE* sind jedoch ebenfalls gültig. Für den body-Parameter ist der `<param-name>` ein JSON-

Name der Eigenschaft	Typ	Beschreibung
		Pfadausdruck ohne das `$.`-Präfix.
`parameter`	`string`	Bei HTTP-APIs sind Anforderungsparameter eine Schlüssel-Wert-Zuordnung, die Parameter angibt, die an `AWS_PROXY` -Integrationen mit einem angegebenen `integrationSubtype` übergeben werden. Sie können statische Werte angeben oder Anforderungsdaten, Stufenvariablen oder Kontextvariablen zuordnen, die zur Laufzeit ausgewertet werden. Weitere Informationen hierzu finden Sie unter the section called “AWS Serviceintegrationen”.

`x-amazon-apigateway-integration.requestParameters`Beispiel für

Im folgenden Beispiel für Anforderungsparameter-Mappings werden die Abfrage- (`version`), Header- (`x-user-id`) und Pfad- (`service`) Parameter einer Methodenanforderung in die Abfrage- (`stage`), Header- (`x-userid`) und Pfad- (op) Parameter der Integrationsanforderung übersetzt.

> **Note**
>
> Wenn Sie Ressourcen über OpenAPI oder erstellen AWS CloudFormation, sollten statische Werte in einfache Anführungszeichen gesetzt werden.
> Um diesen Wert über die Konsole hinzuzufügen, geben Sie `application/json` ohne Anführungszeichen in das Feld ein.

```
"requestParameters" : {
    "integration.request.querystring.stage" : "method.request.querystring.version",
    "integration.request.header.x-userid" : "method.request.header.x-user-id",
    "integration.request.path.op" : "method.request.path.service"
},
```

x-amazon-apigateway-integration.responses-Objekt

Definiert die Antworten der Methode und gibt Parameter-Mappings oder Nutzlast-Mappings von Integrationsantworten auf Methodenantworten an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
Antwortstatusmuster	x-amazon-apigateway-integration.response-Objekt	Entweder ein regulärer Ausdruck, der verwendet wird, um die Integrationsantwort mit der Methodenantwort abzugleichen, oder `default`, um eine Antwort zu erfassen, die noch nicht konfiguriert wurde. Für HTTP-Integrationen gilt der Regex für den Integrationsantwort-Statuscode. Bei Lambda-Aufrufen gilt der reguläre Ausdruck für das `errorMessage` Feld des Fehlerinformationsobjekts, das AWS Lambda als Fehlerantworttext zurückgegeben wird, wenn die Ausführung der Lambda-Funktion eine Ausnahme auslöst.

Name der Eigenschaft	Typ	Beschreibung
		Note Der *Antwortstatusmuster* - Eigenschaftsname bezieht sich auf einen Antwort-Statuscode oder regulären Ausdruck, der eine Gruppe an Antwort-Statuscodes beschreibt. Es entspricht keinem Bezeichner einer IntegrationResponseRessource in der API Gateway REST API.

`x-amazon-apigateway-integration.responses`Beispiel für

Im folgenden Beispiel wird eine Liste mit Antworten von `2xx`- und `302`-Antworten gezeigt. Für die `2xx`-Antwort wird die Methodenantwort von der Nutzlast der Integrationsantwort des `application/json`- oder `application/xml`-MIME-Typs zugeordnet. Diese Antwort verwendet die bereitgestellten Mapping-Vorlagen. Für die `302`-Antwort gibt die Methodenantwort einen `Location`-Header zurück, dessen Wert von der `redirect.url`-Eigenschaft auf der Nutzlast der Integrationsantwort abgerufen wird.

```
"responses" : {
    "2\\d{2}" : {
        "statusCode" : "200",
        "responseTemplates" : {
            "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
            "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
```

```
        }
    },
    "302" : {
        "statusCode" : "302",
        "responseParameters" : {
            "method.response.header.Location": "integration.response.body.redirect.url"
        }
    }
}
```

x-amazon-apigateway-integration.response-Objekt

Definiert eine Antwort und gibt Parameter-Mappings oder Nutzlast-Mappings aus der Integrationsantwort in die Methodenantwort an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`statusCode`	`string`	HTTP-Statuscode für die Methodenantwort, zum Beispie, `"200"`. Dieser muss einer übereinstimmenden Antwort im OpenAPI-Operation `responses` -Feld entsprechen.
`responseTemplates`	x-amazon-apigateway-integration.responseTemplates-Objekt	Gibt MIME-typenspezifische Mapping-Vorlagen für die Nutzlast der Antwort an.
`responseParameters`	x-amazon-apigateway-integration.responseParameters-Objekt	Gibt Parameter-Mappings für die Antwort an. Nur die `header`- und `body`-Parameter der Integrationsantwort können den `header`-

Name der Eigenschaft	Typ	Beschreibung
		Parametern der Methode zugeordnet werden.
`contentHandling`	`string`	Umwandlungstypen für Antwortnutzlastenkodierung. Gültige Werte sind 1) `CONVERT_TO_TEXT` für die Umwandlung einer binären Nutzlast in eine base64-kodierte Zeichenfolge oder die Umwandlung einer Textnutzlast in eine `utf-8`-kodierte Zeichenfolge oder die native Weiterleitung der Textnutzlast ohne Änderung und 2) `CONVERT_TO_BINARY` für die Umwandlung einer Textnutzlast in einen base64-dekodierten Blob oder die native Weiterleitung einer binären Nutzlast ohne Änderung.

`x-amazon-apigateway-integration.response`Beispiel für

Im folgenden Beispiel wird eine `302`-Antwort für die Methode definiert, die eine Nutzlast des `application/json`- oder `application/xml`-MIME-Typs vom Backend ableitet. Die Antwort verwendet die bereitgestellten Mapping-Vorlagen und gibt die Umleitungs-URL von der Integrationsantwort im `Location`-Header der Methode zurück.

```
{
    "statusCode" : "302",
    "responseTemplates" : {
         "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name
\", \"user-id\": \"$root.key\" }",
```

```
          "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
    },
    "responseParameters" : {
        "method.response.header.Location": "integration.response.body.redirect.url"
    }
}
```

x-amazon-apigateway-integration.responseTemplates-Objekt

Gibt die Mapping-Vorlagen für eine Antwortnutzlast der angegebenen MIME-Typen an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
MIME type	`string`	Gibt eine Mapping-Vorlage an, um den Integrationsantwortt ext für einen bestimmten MIME-Typ in den Methodenantworttext umzuwandeln. Weitere Informationen zum Erstellen einer Mapping-Vorlage finden Sie unter PetStore Vorlage für eine Zuordnung. Ein Beispiel für den *MIME-Typ* ist `application/json` .

x-amazon-apigateway-integration.responseTemplate-Beispiel

Im folgenden Beispiel werden Mapping-Vorlagen für eine Anforderungsnutzlast der `application/json`- und `application/xml`-MIME-Typen festgelegt.

```
"responseTemplates" : {
```

```
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration.responseParameters-Objekt

Gibt Mappings von Integrationsmethoden-Antwortparameter auf Methodenantwortparameter an. Sie können `header`, `body` oder statische Werte dem `header`-Typ der Methodenantwort zuordnen. Wird nur für REST-APIs unterstützt.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`method.response.header. <param-name>`	`string`	Der benannte Parameterwert kann von den `header`- und `body`-Typen der Integrationsantwortparameter abgeleitet werden.

`x-amazon-apigateway-integration.responseParameters`Beispiel für

Im folgenden Beispiel werden der `body`- und `header`-Parameter der Integrationsantwort auf zwei `header`-Parameter der Methodenantwort zugewiesen.

```
"responseParameters" : {
    "method.response.header.Location" : "integration.response.body.redirect.url",
    "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

x-amazon-apigateway-integration.tlsConfig-Objekt

Gibt die TLS-Konfiguration für eine Integration an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`insecureSkipVerification`	`Boolean`	Wird nur für REST-APIs unterstützt. Gibt an, ob API Gateway die Überprüfung der Ausstellung des Zertifikats für einen Integrationsendpunkt durch eine unterstützte Zertifizierungsstelle übersprin gt oder nicht. Dies wird nicht empfohlen, ermöglicht Ihnen aber, Zertifikate zu verwenden , die von privaten Zertifizierungsstellen signiert sind, oder Zertifikate, die selbstsig niert sind. Wenn diese Option aktiviert ist, führt API Gateway dennoch eine grundlegende Zertifikatüberprüfung durch, wobei Ablaufdatum, Hostname und das Vorhandensein einer Stammzertifizierungsstelle des Zertifikats geprüft werden. Das Stammzertifikat der privaten Behörde muss die folgenden Einschränkungen erfüllen: • Die x509-Erweiterung `keyUsage` muss `keyCertSign` haben. • Die x509-Erweiterung `basicConstraints` muss `CA:TRUE` haben.

Name der Eigenschaft	Typ	Beschreibung
		Wird nur für `HTTP`- und `HTTP_PROXY` -Integrationen unterstützt. **Warning** Aktivieren von `insecureSkipVerification` wird nicht empfohlen, insbesondere für Integrationen mit öffentlichen HTTPS-Endpunkten. Wenn Sie diese Option aktivieren`insecureSkipVerification`, erhöhen Sie das Risiko von Angriffen. man-in-the-middle

Name der Eigenschaft	Typ	Beschreibung
`serverNameToVerify`	`string`	Wird nur für private HTTP-API-Integrationen unterstützt. Wenn Sie einen Servernamen angeben, verwendet API Gateway diesen, um den Hostnamen auf dem Integrationszertifikat zu überprüfen. Der Servername ist auch im TLS-Handshake enthalten, um Server Name Indication (SNI, Servernamenanzeige) oder virtuelles Hosting zu unterstützen.

x-amazon-apigateway-integration.tlsConfig-Beispiele

Das folgende OpenAPI 3.0-Beispiel aktiviert `insecureSkipVerification` für eine REST-API HTTP-Proxy-Integration.

```
"x-amazon-apigateway-integration": {
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
  "responses": {
     default": {
       "statusCode": "200"
      }
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "ANY",
  "tlsConfig" : {
    "insecureSkipVerification" : true
  }
  "type": "http_proxy",
}
```

Das folgende OpenAPI 3.0-Beispiel gibt einen `serverNameToVerify` für eine private HTTP-API-Integration an.

```
"x-amazon-apigateway-integration" : {
  "payloadFormatVersion" : "1.0",
  "connectionId" : "abc123",
  "type" : "http_proxy",
  "httpMethod" : "ANY",
  "uri" : "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
  "connectionType" : "VPC_LINK",
  "tlsConfig" : {
     "serverNameToVerify" : "example.com"
  }
}
```

x-amazon-apigateway-minimum-Kompressionsgröße

Gibt die minimale Komprimierungsgröße für eine REST-API an. Um die Komprimierung zu aktivieren, geben Sie eine ganze Zahl zwischen 0 und 10485760 an. Weitere Informationen hierzu finden Sie unter Aktivieren der Nutzlastkomprimierung für eine API.

x-amazon-apigateway-minimumBeispiel für eine Kompressionsgröße

Das folgende Beispiel gibt eine minimale Komprimierungsgröße von `5242880` Bytes für eine REST-API an.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Gibt eine Ressourcen-Richtlinie für eine REST-API an. Weitere Informationen zu den Ressourcenrichtlinien finden Sie unter Zugriff auf eine API mit API Gateway-Ressourcenrichtlinien steuern. Beispiele für Ressourcenrichtlinien finden Sie unter Beispiele für API Gateway-Ressourcenrichtlinien.

`x-amazon-apigateway-policy`Beispiel für

Das folgende Beispiel gibt eine Ressourcen-Richtlinie für eine REST-API an. Die Ressourcenrichtlinie verweigert (blockiert) eingehenden Datenverkehr von einem angegebenen Quell-IP-Adressblock an eine API. Wird beim Import unter Verwendung der aktuellen Region in Ihre AWS Konto-

ID und die aktuelle REST-API-ID konvertiert. `"execute-api:/*"` `arn:aws:execute-api:region:account-id:api-id/*`

```
"x-amazon-apigateway-policy": {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

x-amazon-apigateway-request-validator-Eigenschaft

Gibt eine Anforderungsvalidierung an, indem auf einen *request_validator_name* der x-amazon-apigateway-request-validators-Objekt-Zuweisung verwiesen wird, um die Anforderungsvalidierung auf der enthaltenen API oder einer Methode zu aktivieren. Der Wert dieser Erweiterung ist eine JSON-Zeichenfolge.

Diese Erweiterung kann auf API-Ebene oder auf Methodenebene angegeben werden. Die Validierung auf API-Ebene gilt für alle Methoden, es sei denn, sie wird von der Validierung auf Methodenebene außer Kraft gesetzt.

x-amazon-apigateway-request-validator Beispiel für

Im folgenden Beispiel wird die `basic`-Anforderungsvalidierung auf API-Ebene angewendet, während die `parameter-only`-Anforderungsvalidierung auf der `POST /validation`-Anforderung angewendet wird.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "basic",
  "paths": {
    "/validation": {
      "post": {
        "x-amazon-apigateway-request-validator" : "params-only",
        ...
     }
}
```

x-amazon-apigateway-request-validators-Objekt

Definiert die unterstützten Anforderungsvalidierungen für die enthaltene API als eine Zuweisung zwischen einem Validierungsnamen und den zugehörigen Anforderungsvalidierungsregeln. Diese Erweiterung gilt für eine REST-API.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
request_validator_name	x-amazon-apigateway-request-Validators.RequestValidator-Objekt	Gibt die Validierungsregeln an, die aus der benannten Validierung bestehen. Zum Beispiel: `"basic" : { "validateRequestBody" : true, "validateRequestParameters" : true },` Um diese Validierung auf eine bestimmte Methode anzuwenden, muss der Validierungsname (`basic`) als der Wert der x-amazon-apigateway-request-validator-Eigenschaft-Eigenschaft referenziert werden.

`x-amazon-apigateway-request-validators`Beispiel für

Das folgende Beispiel zeigt eine Reihe von Anforderungsvalidierungen für eine API als eine Zuweisung zwischen einem Validierungsnamen und den zugehörigen Anforderungsvalidierungsregeln.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
```

```
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  ...
}
```

x-amazon-apigateway-request-Validators.RequestValidator-Objekt

Gibt die Validierungsregeln einer Anforderungsvalidierung als Teil der x-amazon-apigateway-request-validators-Objekt-Zuweisungsdefinition an.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`validateRequestBody`	`Boolean`	Gibt an, ob der Anforderungstext (`true`) oder nicht (`false`) validiert werden soll.
`validateRequestParameters`	`Boolean`	Gibt an, ob die erforderlichen Anforderungsparameter (`true`) oder nicht (`false`) validiert werden.

x-amazon-apigateway-request-validators.requestValidator Beispiel für

Das folgende Beispiel zeigt eine Anforderungsvalidierung nur für den Parameter:

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

x-amazon-apigateway-tag-value-Eigenschaft

Gibt den Wert eines AWS -Tags für eine HTTP-API an. Sie können die `x-amazon-apigateway-tag-value` Eigenschaft als Teil des OpenAPI-Tag-Objekts auf Stammebene verwenden, um AWS Tags für eine HTTP-API anzugeben. Wenn Sie einen Tag-Namen ohne die `x-amazon-apigateway-tag-value`-Eigenschaft angeben, erstellt API Gateway für einen Wert ein Tag mit einer leeren Zeichenfolge.

Weitere Informationen zum Taggen finden Sie unter API Gateway-Ressourcen taggen.

Eigenschaften

Name der Eigenschaft	Typ	Beschreibung
`name`	`String`	Gibt den Tag-Schlüssel an.
`x-amazon-apigateway-tag-value`	`String`	Gibt den Tag-Wert an.

`x-amazon-apigateway-tag-value`Beispiel für

Das folgende Beispiel gibt zwei Tags für eine HTTP-API an:

- "Owner": "Admin"
- "Prod": ""

```
"tags": [
    {
      "name": "Owner",
      "x-amazon-apigateway-tag-value": "Admin"
    },
    {
      "name": "Prod"
    }
]
```

Sicherheit in Amazon API Gateway

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der übergreifenden Verantwortlichkeit beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für Amazon API Gateway gelten, finden Sie unter AWS Services im Umfang nach Compliance-Programm AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation erläutert, wie das Modell der geteilten Verantwortung bei der Verwendung von API Gateway angewendet werden kann. Die folgenden Themen zeigen Ihnen, wie Sie API Gateway konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS Dienste verwenden können, die Ihnen helfen, Ihre API-Gateway-Ressourcen zu überwachen und zu sichern.

Weitere Informationen finden Sie unter Sicherheit in Amazon API Gateway.

Themen

- Bewährte Methoden in Amazon API Gateway

Datenschutz in Amazon API Gateway

Das AWSModell der geteilten Verantwortung wird auch auf den Datenschutz in Amazon API Gateway angewendet. Wie in diesem Modell beschrieben, ist AWS verantwortlich für den Schutz der globalen Infrastruktur, in der die gesamte AWS Cloud ausgeführt wird. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS-Modell der geteilten Verantwortung und in der DSGVO im AWS-Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, AWS-Konto-Anmeldeinformationen zu schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit AWS CloudTrail ein.
- Verwenden Sie AWS-Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit API Gateway oder anderen AWS-Services über die Konsole, API, AWS CLI oder AWS SDKs arbeiten. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können

für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung in Amazon API Gateway

Datenschutz bezieht sich auf den Schutz von Daten während der Übertragung (auf dem Weg zu und von API Gateway) und im Ruhezustand (während sie in der gespeichert werde AWS).

Datenverschlüsselung im Ruhezustand in Amazon API Gateway

Wenn Sie das Caching für eine REST-API aktivieren, können Sie die Cache-Verschlüsselung aktivieren. Weitere Informationen hierzu finden Sie unter Aktivieren von API-Caching für verbesserte Reaktionsfähigkeit.

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag AWS Shared Responsibility Model and GDPR im AWS-Sicherheitsblog.

Datenverschlüsselung während der Übertragung in Amazon API Gateway

Die mit Amazon API Gateway erstellten APIs stellen HTTPS-Endpunkte dar. API Gateway unterstützt keine unverschlüsselten Endpunkte (HTTP).

API Gateway verwaltet die Zertifikate für `execute-api`-Standardendpunkte. Wenn Sie einen benutzerdefinierten Domainnamen konfigurieren, geben Sie das Zertifikat für den Domainnamen an. Als bewährte Methode sollten Sie keine PIN-Zertifikate verwenden.

Für mehr Sicherheit können Sie eine Mindestversion des TLS-Protokolls (Transport Layer Security) auswählen, die für Ihre benutzerdefinierte API Gateway-Domäne durchgesetzt wird. WebSocket-APIs und HTTP-APIs unterstützen nur TLS 1.2. Weitere Informationen hierzu finden Sie unter Auswahl einer Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway.

Sie können außerdem eine Amazon CloudFront-Distribution mit einem benutzerdefinierten SSL-Zertifikat in Ihrem Konto einrichten und diese mit regionalen APIs verwenden. Anschließend können Sie die Sicherheitsrichtlinie für die CloudFront-Verteilung mit TLS 1.1 oder höher entsprechend Ihren Sicherheits- und Compliance-Anfragen konfigurieren.

Weitere Informationen zum Datenschutz enthält REST-API schützen und der Blog-Beitrag AWS Shared Responsibility Model and GDPR im AWS-Sicherheitsblog.

Richtlinie für den Datenverkehr zwischen Netzwerken

Mit Amazon API Gateway können Sie private REST-APIs erstellen, auf die nur von Ihrer Amazon Virtual Private Cloud (VPC) aus zugegriffen werden kann. Die VPC verwendet einen Schnittstellen-VPC-Endpunkt, bei dem es sich um eine Endpunkt-Netzwerkschnittstelle handelt, die Sie in Ihrer VPC erstellen. Mithilfe von Ressourcen-Richtlinien können Sie den Zugriff auf Ihre API von aus ausgewählten VPCs und VPC-Endpunkten erlauben oder verweigern, darunter auch über AWS-Konten. Jeder Endpunkt kann für den Zugriff auf mehrere APIs verwendet werden. Sie können mit AWS Direct Connect auch eine Verbindung von einem On-Premises-Netzwerk zu Amazon VPC herstellen und über diese Verbindung auf Ihre private API zugreifen. In allen Fällen verwendet der Datenverkehr zu Ihrer privaten API eine sichere Verbindung und verlässt nicht das Amazon-Netzwerk. Er ist vom öffentlichen Internet isoliert. Weitere Informationen hierzu finden Sie unter the section called “Private REST-APIs”.

Identitäts- und Zugriffsverwaltung für Amazon API Gateway

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem ein Administrator den Zugriff auf AWS-Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer authentifiziert werden kann (d. h. wer sich anmelden kann) und wer autorisiert werden kann (d. h. wer Berechtigungen erhalten kann), um API Gateway-Ressourcen zu nutzen. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Die Art der Verwendung von AWS Identity and Access Management (IAM) ist von den Aktionen abhängig, die Sie in API Gateway ausführen.

Service-Benutzer – Wenn Sie den API Gateway-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere API Gateway-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle verstehen, kann Ihnen dies helfen, die richtigen Berechtigungen von Ihrem Administrator anzufordern. Wenn Sie nicht auf eine Funktion in API Gateway zugreifen können, informieren Sie sich in Fehlerbehebung bei Amazon API Gateway-Identität und -Zugriff.

Service-Administrator – Wenn Sie in Ihrem Unternehmen die Verantwortung für API Gateway-Ressourcen haben, haben Sie wahrscheinlich vollständigen Zugriff auf API Gateway. Sie legen die Funktionen und Ressourcen von API Gateway fest, auf die Mitarbeiter zugreifen können. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM zu verstehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit API Gateway verwenden kann, finden Sie unter Funktionsweise von Amazon API Gateway mit IAM.

IAM-Administrator – Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf API Gateway verfassen können. Beispiele für identitätsbasierte API Gateway-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien.

Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center. (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter Anmelden bei Ihrem AWS-Konto im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuerten Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anfragen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS empfiehlt beispielsweise die Verwendung von Multi-Faktor Authentifizierung (MFA), um die Sicherheit Ihres Kontos zu verbessern. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center-Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto-Stammbenutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges

Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM roles (IAM-Rollen)

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wenn eine Verbundidentität authentifiziert wird, wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center-Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto

zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff – Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Service kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward access sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anforderungen werden nur dann gestellt, wenn ein Service eine Anforderung erhält, die Interaktionen mit anderen AWS-Services oder Ressourcen erfordert, um abgeschlossen werden zu können. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Forward Access Sessions (FAS).
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Serviceverbundene Rolle – Eine serviceverbundene Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen in Amazon EC2 – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und AWS CLI- oder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die

Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole` -Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern,

welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem AWS-Konto anfügen können. Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Service. Sie können verwaltete AWS-Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3, AWS WAF und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. Weitere Informationen zu ACLs finden Sie unter Zugriffssteuerungsliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Ein ausdrückliches Ablehnen in einer dieser Richtlinien setzt das Zulassen außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Service für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. SCPs schränken Berechtigungen für Entitäten in Mitgliedskonten einschließlich des jeweiligen Root-Benutzer des AWS-Kontos ein. Weitere Informationen zu Organizations und SCPs finden Sie unter Featuresweise von SCPs im AWS Organizations-Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Funktionsweise von Amazon API Gateway mit IAM

Bevor Sie mit IAM den Zugriff auf API Gateway verwalten können, sollten Sie sich darüber informieren, welche IAM-Funktionen Sie mit API Gateway verwenden können. Einen Überblick über die Zusammenarbeit von API Gateway und anderen AWS-Services mit IAM finden Sie unter AWS-Services, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Themen

- Identitätsbasierte API-Gateway-Richtlinien
- Ressourcenbasierte API Gateway-Richtlinien
- Autorisierung basierend auf API Gateway-Tags
- API Gateway-IAM-Rollen

Identitätsbasierte API-Gateway-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. API Gateway unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Weitere Informationen zu den API-Gateway-spezifischen Aktionen, Ressourcen und Bedingungsschlüsseln finden Sie im Abschnitt Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management und Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management V2. Informationen zu allen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Das folgende Beispiel zeigt eine identitätsbasierte Richtlinie, die es einem Benutzer ermöglicht, nur private REST-APIs zu erstellen oder zu aktualisieren. Weitere Beispiele finden Sie unter the section called “Beispiele für identitätsbasierte Richtlinien”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
```

```
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/??????????"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/EndpointType": "PRIVATE",
          "apigateway:Resource/EndpointType": "PRIVATE"
        }
      }
    },
    {
      "Sid": "AllowResourcePolicyUpdates",
      "Effect": "Allow",
      "Action": [
          "apigateway:UpdateRestApiPolicy"
      ],
      "Resource": [
          "arn:aws:apigateway:us-east-1::/restapis/*"
      ]
    }
  ]
}
```

Aktionen

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie in einer Richtlinie den Zugriff erlauben oder verweigern können.

Richtlinienaktionen in API Gateway verwenden das folgende Präfix vor der Aktion: `apigateway:`. Richtlinienanweisungen müssen entweder ein `Action`- oder ein `NotAction`-Element enthalten. API Gateway definiert eine eigene Gruppe von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen können.

Der API-verwaltende `Action`-Ausdruck hat das Format `apigateway:`*`action`*, wobei *`action`* eine der folgenden API-Gateway-Aktionen ist: GET, POST, PUT, DELETE, PATCH (zur Aktualisierung von Ressourcen) oder * für alle vorherigen Aktionen.

Einige Beispiele für den `Action`-Ausdruck sind:

- **`apigateway:*`** für alle API Gateway-Aktionen.
- **`apigateway:GET`** nur für die GET-Aktion in API Gateway.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie folgendermaßen mit Kommas:

```
"Action": [
      "apigateway:action1",
      "apigateway:action2"
```

Informationen zu HTTP-Verben für bestimmte API-Gateway-Vorgänge finden Sie unter Amazon API Gateway Version 1 API-Referenz (REST-APIs) und Amazon API Gateway Version 2 API-Referenz (WebSocket und HTTP-APIs).

Weitere Informationen finden Sie unter the section called “Beispiele für identitätsbasierte Richtlinien”.

Ressourcen

Administratoren können mit AWS-JSON-Richtlinien festlegen, welche Personen zum Zugriff auf welche Ressourcen berechtigt sind. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein `Resource`- oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

API Gateway-Ressourcen verfügen über das folgende ARN-Format:

```
arn:aws:apigateway:region::resource-path-specifier
```

Um beispielsweise eine REST-API mit der ID *api-id* und ihren Unterressourcen, wie Autorisierern in Ihrer Anweisung, anzugeben, verwenden Sie den folgenden ARN:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Um alle REST-APIs und Unterressourcen anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Eine Liste der API Gateway-Ressourcentypen und ihrer ARNs finden Sie unter Referenz zu API Gateway Amazon-Ressourcenname (ARN).

Bedingungsschlüssel

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die Bedingung mittels einer logischen OR-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Eine Liste aller globalen AWS-Bedingungsschlüssel finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

API Gateway definiert einen eigenen Satz von Bedingungsschlüsseln und unterstützt auch einige globale Bedingungsschlüssel. Eine Liste der API Gateway-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon API Gateway im IAM-Benutzerhandbuch. Informationen zu den Aktionen und Ressourcen, die Sie mit einem Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon API Gateway definierte Aktionen.

Informationen zur Markierung, einschließlich attributbasierter Zugriffskontrolle, finden Sie unter Markieren.

Beispiele

Beispiele für identitätsbasierte API Gateway-Richtlinien finden Sie unter Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien.

Ressourcenbasierte API Gateway-Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die angeben, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für die API Gateway-Ressource ausführen kann. API Gateway unterstützt ressourcenbasierte Berechtigungsrichtlinien für REST-APIs. Sie verwenden Ressourcenrichtlinien, um zu steuern, wer eine REST-API aufrufen kann. Weitere Informationen finden Sie unter the section called “API Gateway-Ressourcenrichtlinien verwenden”.

Beispiele

Beispiele für ressourcenbasierte API Gateway-Richtlinien finden Sie unter Beispiele für API Gateway-Ressourcenrichtlinien.

Autorisierung basierend auf API Gateway-Tags

Sie können Tags an API Gateway-Ressourcen anfügen oder Tags in einer Anforderung an API Gateway übergeben. Um den Zugriff basierend auf Tags zu steuern, stellen Sie Tag-Informationen im Bedingungselement einer Richtlinie unter Verwendung der Bedingungsschlüssel `apigateway:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* oder `aws:TagKeys` bereit. Weitere Informationen über das Markieren mit Tags von API Gateway-Ressourcen finden Sie unter the section called “Attributbasierte Zugriffskontrolle”.

Beispiele für identitätsbasierte Richtlinien zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Tags dieser Ressource finden Sie unter Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden.

API Gateway-IAM-Rollen

Eine IAM-Rolle ist eine Entität in Ihrem AWS-Konto mit spezifischen Berechtigungen.

Verwenden temporärer Anmeldeinformationen mit API Gateway

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Temporäre Sicherheitsanmeldeinformationen erhalten Sie durch Aufrufen von AWS STS-API-Vorgängen wie AssumeRole oder GetFederationToken.

API Gateway unterstützt die Verwendung temporärer Anmeldeinformationen.

Serviceverknüpfte Rollen

Serviceverknüpfte Rollen erlauben AWS-Services den Zugriff auf Ressourcen in anderen Services, um eine Aktion in Ihrem Auftrag auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

API Gateway unterstützt serviceverknüpfte Rollen. Informationen zum Erstellen oder Verwalten von serviceverknüpften API Gateway-Rollen finden Sie unter Verwenden von serviceverknüpften Rollen für API Gateway.

Servicerollen

Ein Service kann in Ihrem Namen eine Service-Rolle annehmen. Eine Service-Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Services, um eine Aktion in Ihrem Namen auszuführen. Service-Rollen werden in Ihrem IAM-Konto angezeigt und gehören dem Konto, sodass ein Administrator die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktionalität des Services beeinträchtigen.

API Gateway unterstützt Servicerollen.

Beispiele für identitätsbasierte Amazon API Gateway-Richtlinien

Standardmäßig verfügen IAM-Benutzer und -Rollen nicht über die Berechtigung zum Erstellen oder Ändern von API Gateway-Ressourcen. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS CLI oder AWS SDKs ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator

muss diese Richtlinien anschließend den IAM-Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen zum Erstellen von IAM-Richtlinien finden Sie unter Erstellen von Richtlinien im JSON-Reiter im IAM-Benutzerhandbuch. Weitere Informationen zu den Aktionen, Ressourcen und Bedingungen, die für API Gateway spezifisch sind, finden Sie im Abschnitt Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management und Aktionen, Ressourcen und Bedingungsschlüssel für Amazon-API-Gateway-Management V2.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand API-Gateway-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS-verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen – Um Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie die AWS-verwaltete Richtlinien die Berechtigungen für viele allgemeine Anwendungsfälle gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie AWS-kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien oder AWS-verwaltete Richtlinien für AuftragsFeatureen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer

Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.

- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn diese durch ein bestimmtes AWS-Service, wie beispielsweise AWS CloudFormation, verwendet werden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und Featureale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und Featureale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Bedarf einer Multi-Faktor-Authentifizierung (MFA) – Wenn Sie ein Szenario haben, das IAM-Benutzer oder Root-Benutzer in Ihrem AWS-Konto erfordert, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die AWS-API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Einfache Leseberechtigungen

Diese Beispielrichtlinie gibt einem Benutzer die Berechtigung, Informationen über alle Ressourcen einer HTTP- oder WebSocket-API mit der Kennung von `a123456789` in der AWS-Region „us-east-1“ zu erhalten. Die Ressource `arn:aws:apigateway:us-east-1::/apis/a123456789/*` umfasst alle Unterressourcen der API, wie Autorisierer und Bereitstellungen.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}
```

Nur REQUEST- oder JWT-Autorisierer erstellen

Diese Beispielrichtlinie ermöglicht es einem Benutzer, APIs nur mit `REQUEST`- oder `JWT`-Autorisierern zu erstellen, einschließlich durch Import. Im `Resource`-Abschnitt der Richtlinie `arn:aws:apigateway:us-east-1::/apis/??????????` ist festgelegt, dass Ressourcen maximal 10 Zeichen haben, was Unterressourcen einer API ausschließt. Dieses Beispiel verwenden `ForAllValues` im `Condition`-Abschnitt, da Benutzer mehrere Autorisierer gleichzeitig erstellen können, indem sie eine API importieren.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
```

```
            "REQUEST",
            "JWT"
          ]
        }
      }
    }
  ]
}
```

Erfordern, dass der **`execute-api`**-Standard-Endpunkt deaktiviert ist

Diese Beispielrichtlinie ermöglicht es Benutzern, eine API mit der Anforderung, dass `DisableExecuteApiEndpoint` `true` ist, zu erstellen, zu aktualisieren oder zu importieren. Wenn `DisableExecuteApiEndpoint` `true` ist, können Clients den `execute-api`-Standard-Endpunkt nicht zum Aufrufen einer API verwenden.

Wir verwenden die `BoolIfExists`-Bedingung, um einen Aufruf zur Aktualisierung einer API zu bearbeiten, für die der `DisableExecuteApiEndpoint`-Bedingungsschlüssel nicht ausgefüllt ist. Wenn ein Benutzer versucht, eine API zu erstellen oder zu importieren, wird der `DisableExecuteApiEndpoint`-Bedingungsschlüssel immer ausgefüllt.

Da die `apis/*`-Ressource auch Unterressourcen, wie Autorisierer oder Methoden, erfasst, haben wir sie explizit nur auf APIs mit einer `Deny`-Anweisung festgelegt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
```

```
        }
      }
    },
    {
      "Sid": "ScopeDownToJustApis",
      "Effect": "Deny",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/*/*"
      ]
    }
  ]
}
```

Benutzern erlauben, nur private REST-APIs zu erstellen oder zu aktualisieren

Diese Beispielrichtlinie verwendet Anforderungsbedingungsschlüssel, um zu verlangen, dass ein Benutzer nur `PRIVATE`-APIs erstellt, und um Updates zu verhindern, die eine API von `PRIVATE` in einen anderen Typ ändern könnten, wie `REGIONAL`.

Wir verwenden `ForAllValues`, um zu verlangen, dass jeder `EndpointType`, der einer API hinzugefügt wird, `PRIVATE` ist. Wir verwenden einen Ressourcenbedingungsschlüssel, um jedes Update für eine API zu ermöglichen, solange es `PRIVATE` ist. `ForAllValues` gilt nur, wenn ein Bedingungsschlüssel vorhanden ist.

Wir verwenden den Non-Greedy Matcher (`?`), um explizit mit API-IDs abzugleichen, um zu verhindern, dass Nicht-API-Ressourcen, wie Autorisierer, zugelassen werden.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScopePutToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT"
            ],
            "Resource": [
```

```
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "ScopeToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Request/EndpointType": "PRIVATE",
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "AllowResourcePolicyUpdates",
            "Effect": "Allow",
            "Action": [
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis/*"
            ]
        }
    ]
}
```

Erfordern, dass API-Routen Autorisierung haben

Diese Richtlinie führt dazu, dass Versuche, eine Route zu erstellen oder zu aktualisieren (einschließlich durch Import), fehlschlagen, wenn die Route keine Autorisierung hat. `ForAnyValue` wird als „false“ ausgewertet, wenn der Schlüssel nicht vorhanden ist, z. B. wenn eine Route nicht erstellt oder aktualisiert wird. Wir verwenden `ForAnyValue`, da durch den Import mehrere Routen erstellt werden können.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "apigateway:Request/RouteAuthorizationType": "NONE"
        }
      }
```

```
    }
  ]
}
```

Verhindern Sie, dass ein Benutzer einen VPC-Link erstellt oder aktualisiert

Diese Richtlinie verhindert, dass ein Benutzer einen VPC-Link erstellt oder aktualisiert. Ein VPC-Link ermöglicht es Ihnen, Ressourcen innerhalb einer Amazon VPC an Clients außerhalb der VPC bereitzustellen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    }
  ]
}
```

Beispiele für ressourcenbasierte Richtlinien für Amazon API Gateway

Beispiele für eine ressourcenbasierte Richtlinie finden Sie unter the section called “Beispiele für API Gateway-Ressourcenrichtlinien”.

Fehlerbehebung bei Amazon API Gateway-Identität und -Zugriff

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von API Gateway und IAM auftreten können.

Themen

- Ich bin nicht zur Ausführung einer Aktion in API Gateway autorisiert

- Ich bin nicht zur Ausführung von iam:PassRole autorisiert
- Ich möchte Personen außerhalb meines AWS-Kontos Zugriff auf meine API-Gateway-Ressourcen erteilen

Ich bin nicht zur Ausführung einer Aktion in API Gateway autorisiert

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `apigateway:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 apigateway:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `apigateway:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht zur Ausführung von iam:PassRole autorisiert

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion „`iam:PassRole`“ autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an API Gateway übergeben zu können.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Service, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Service.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in API Gateway auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Kontos Zugriff auf meine API-Gateway-Ressourcen erteilen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Services, die ressourcenbasierte Richtlinien oder Zugriffssteuerungslisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen finden Sie hier:

- Informationen dazu, ob API Gateway diese Funktionen unterstützt, finden Sie unter Funktionsweise von Amazon API Gateway mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Verwenden von serviceverknüpften Rollen für API Gateway

Amazon API Gateway verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen . Eine serviceverknüpfte Rolle ist ein spezieller Typ einer IAM-Rolle, die direkt mit API Gateway verknüpft ist. Serviceverknüpfte Rollen werden von API Gateway vordefiniert und schließen

alle Berechtigungen ein, die der Service zum Aufrufen anderer - AWS Services in Ihrem Namen erfordert.

Eine serviceverknüpfte Rolle vereinfacht die Einrichtung von API Gateway, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. API Gateway definiert die Berechtigungen seiner serviceverknüpften Rollen. Sofern keine andere Konfiguration festgelegt wurde, kann nur API Gateway die Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem die zugehörigen Ressourcen gelöscht wurden. Dies schützt Ihre API Gateway-Ressourcen, da Sie nicht versehentlich die Berechtigung für den Zugriff auf die Ressourcen entfernen können.

Informationen zu anderen Services, die serviceverknüpfte Rollen unterstützen, finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Berechtigungen von serviceverknüpften Rollen für API Gateway

API Gateway verwendet die serviceverknüpfte Rolle namens AWSServiceRoleForAPIGateway – Ermöglicht API Gateway den Zugriff auf Elastic Load Balancing , Amazon Data Firehose und andere Serviceressourcen in Ihrem Namen.

Die AWSServiceRoleForAPIGateway serviceverknüpfte Rolle vertraut darauf, dass die folgenden Services die Rolle übernehmen:

- `ops.apigateway.amazonaws.com`

Die Rollenberechtigungsrichtlinie erlaubt API Gateway die Durchführung der folgenden Aktionen für die angegebenen Ressourcen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
```

```
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingTargets",
                "xray:GetSamplingRules",
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "servicediscovery:DiscoverInstances"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:GetCertificate"
            ],
            "Resource": "arn:aws:acm:*:*:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
```

```
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Owner",
                        "VpcLinkId"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeVpcs",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetNamespace",
            "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetService",
            "Resource": "arn:aws:servicediscovery:*:*:service/*"
        }
    ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigungen im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpften Rolle für API Gateway

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine API, einen benutzerdefinierten Domänennamen oder einen VPC-Link in der AWS CLI, AWS Management Console der oder der AWS API erstellen, erstellt API Gateway die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie eine API, einen benutzerdefinierten Domänennamen oder einen VPC-Link erstellen, erstellt API Gateway die serviceverknüpfte Rolle erneut für Sie.

Bearbeiten einer serviceverknüpften Rolle für API Gateway

API Gateway erlaubt es Ihnen nicht, die AWSServiceRoleForAPIGateway serviceverknüpfte Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach der Erstellung einer serviceverknüpften Rolle nicht bearbeitet werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für API Gateway

Wenn Sie eine Funktion oder einen Service, die bzw. der eine serviceverknüpfte Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpfte Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

> **Note**
>
> Wenn der API Gateway-Service die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

So löschen Sie API Gateway-Ressourcen, die von der verwendet werden AWSServiceRoleForAPIGateway

1. Öffnen Sie die API Gateway-Konsole unter https://console.aws.amazon.com/apigateway/.
2. Navigieren Sie zu der API, dem benutzerdefinierten Domänennamen oder dem VPC-Link, der die serviceverknüpfte Rolle verwendet.
3. Verwenden Sie die Konsole, um die Ressource zu löschen.
4. Wiederholen Sie den Vorgang, um alle APIs, benutzerdefinierten Domänennamen oder VPC-Links zu löschen, die die serviceverknüpfte Rolle verwenden.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die - AWS API AWS CLI, um die AWSServiceRoleForAPIGateway serviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Unterstützte Regionen für serviceverknüpfte API Gateway-Rollen

API Gateway unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter AWS -Service-Endpunkte.

API Gateway-Updates für - AWS verwaltete Richtlinien

Anzeigen von Details zu Aktualisierungen für - AWS verwaltete Richtlinien für API Gateway, seit dieser Service mit der Verfolgung dieser Änderungen begonnen hat. Um automatische Warnungen über Änderungen an dieser Seite erhalten, abonnieren Sie den RSS-Feed auf der API-Gateway-Dokumentverlauf-Seite.

Änderungen	Beschreibung	Datum
`acm:GetCertificate` - Unterstützung für `AWSServiceRoleForAPIGateway` - Richtlinien wurde hinzugefügt.	Die `AWSServiceRoleForAPIGateway` -Richtlinie enthält jetzt die Berechtigung zum Aufrufen des ACM-`GetCertificate` API-Aktion	12. Juli 2021

Änderungen	Beschreibung	Datum
API Gateway hat mit der Verfolgung von Änderungen begonnen	API Gateway hat mit der Verfolgung von Änderungen für seine AWS -verwalteten Richtlinien begonnen.	12. Juli 2021

Protokollierung und Überwachung in Amazon API Gateway

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von API Gateway und Ihren AWS Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen Fehler an mehreren Stellen leichter debuggen können, falls einer auftritt. AWS bietet verschiedene Tools zur Überwachung Ihrer API-Gateway-Ressourcen und zur Reaktion auf potenzielle Vorfälle:

CloudWatch Amazon-Protokolle

Um Probleme im Zusammenhang mit der Ausführung von Anfragen oder dem Client-Zugriff auf Ihre API zu beheben, können Sie CloudWatch Logs aktivieren, um API-Aufrufe zu protokollieren. Weitere Informationen finden Sie unter the section called “CloudWatch Logs”.

CloudWatch Amazon-Alarme

Mithilfe von CloudWatch Alarmen beobachten Sie eine einzelne Metrik über einen von Ihnen festgelegten Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird eine Benachrichtigung an ein Thema oder eine AWS Auto Scaling Richtlinie von Amazon Simple Notification Service gesendet. CloudWatch Alarme lösen keine Aktionen aus, wenn sich eine Metrik in einem bestimmten Status befindet. Der Status muss sich stattdessen geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein. Weitere Informationen finden Sie unter the section called “CloudWatch Metriken”.

Zugriffs-Logging auf Firehose

Um Probleme im Zusammenhang mit dem Client-Zugriff auf Ihre API zu beheben, können Sie Firehose so einrichten, dass API-Aufrufe protokolliert werden. Weitere Informationen finden Sie unter the section called “Firehose”.

AWS CloudTrail

CloudTrail bietet eine Aufzeichnung der Aktionen, die von einem Benutzer, einer Rolle oder einem AWS Dienst in API Gateway ausgeführt wurden. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an API Gateway gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln. Weitere Informationen finden Sie unter the section called “Arbeiten mit CloudTrail”.

AWS X-Ray

X-Ray ist ein AWS Dienst, der Daten über die Anfragen sammelt, die Ihre Anwendung bedient, und anhand dieser Daten eine Service-Map erstellt, anhand derer Sie Probleme mit Ihrer Anwendung und Optimierungsmöglichkeiten identifizieren können. Weitere Informationen finden Sie unter the section called “Einrichten AWS X-Ray”.

AWS Config

AWS Config bietet einen detaillierten Überblick über die Konfiguration der AWS Ressourcen in Ihrem Konto. Sie können Ressourcenbeziehungen betrachten, einen Verlauf der Konfigurationsänderungen anzeigen und feststellen, wie sich Beziehungen und Konfigurationen im Lauf der Zeit ändern. Sie können AWS Config damit Regeln definieren, mit denen Ressourcenkonfigurationen im Hinblick auf Datenkonformität bewertet werden. AWS Config Regeln stellen die idealen Konfigurationseinstellungen für Ihre API-Gateway-Ressourcen dar. Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet wird, AWS Config kann ich Sie mithilfe eines Amazon Simple Notification Service (Amazon SNS) - Themas benachrichtigen. Details hierzu finden Sie unter the section called “Arbeiten mit AWS Config”.

Protokollieren Amazon API Gateway Gateway-API-Aufrufen mit AWS CloudTrail

Amazon API Gateway ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle REST-API-Aufrufe für den API Gateway Gateway-Dienst als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der API Gateway Gateway-Konsole und Codeaufrufen an die API Gateway Gateway-Dienst-APIs. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an API Gateway gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wann sie gestellt wurde, und weitere Details ermitteln.

> **Note**
>
> TestInvokeAuthorizerund TestInvokeMethodsind nicht angemeldet CloudTrail.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Ob die Anfrage im Namen eines IAM Identity Center-Benutzers gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS Management Console sind regionsübergreifend. Sie können einen Pfad mit einer oder mehreren Regionen erstellen, indem Sie den verwenden. AWS CLI Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten in Ihrem Konto AWS-Regionen erfassen. Wenn du einen Trail mit nur einer Region erstellst, kannst du dir nur die Ereignisse ansehen, die in den Trails protokolliert wurden. AWS-Region Weitere Informationen zu Trails finden Sie unter Einen Trail für Sie erstellen AWS-Konto und Einen Trail für eine Organisation erstellen im AWS CloudTrail Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-

Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3-Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur Preisgestaltung finden Sie unter CloudTrail AWS CloudTrail Preisgestaltung.

API Gateway Gateway-Verwaltungsereignisse in CloudTrail

Verwaltungsereignisse bieten Informationen zu Verwaltungsvorgängen, die an Ressourcen in Ihrem ausgeführt werden AWS-Konto. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. In der Standardeinstellung werden Verwaltungsereignisse CloudTrail protokolliert.

Amazon API Gateway protokolliert alle API Gateway Gateway-Aktionen als Verwaltungsereignisse, mit Ausnahme von TestInvokeAuthorizerund TestInvokeMethod. Eine Liste der Amazon API Gateway-Aktionen, bei denen sich API Gateway anmeldet CloudTrail, finden Sie in der Amazon API Gateway API-Referenz.

Beispiel für ein API-Gateway-Ereignis

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das die `GetResource` API-Gateway-Aktion demonstriert:


```
{
    Records: [
        {
            eventVersion: "1.03",
            userIdentity: {
                type: "Root",
                principalId: "AKIAI44QH8DHBEXAMPLE",
                arn: "arn:aws:iam::123456789012:root",
                accountId: "123456789012",
                accessKeyId: "AKIAIOSFODNN7EXAMPLE",
                sessionContext: {
                    attributes: {
                        mfaAuthenticated: "false",
                        creationDate: "2015-06-16T23:37:58Z"
                    }
                }
            },
            eventTime: "2015-06-17T00:47:28Z",
            eventSource: "apigateway.amazonaws.com",
            eventName: "GetResource",
            awsRegion: "us-east-1",
            sourceIPAddress: "203.0.113.11",
            userAgent: "example-user-agent-string",
            requestParameters: {
                restApiId: "3rbEXAMPLE",
                resourceId: "5tfEXAMPLE",
                template: false
            },
            responseElements: null,
            requestID: "6d9c4bfc-148a-11e5-81b6-7577cEXAMPLE",
            eventID: "4d293154-a15b-4c33-9e0a-ff5eeEXAMPLE",
            readOnly: true,
            eventType: "AwsApiCall",
            recipientAccountId: "123456789012"
        },
        ... additional entries ...
    ]
}
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalte.

API Gateway API-Konfiguration mit überwache AWS Config

Sie können mit AWS Config Konfigurationsänderungen an Ihren API-Gateway-API-Ressourcen aufzeichnen und Benachrichtigungen auf der Grundlage von Ressourcenänderungen senden. Die Pflege eines Konfigurations-Änderungsverlaufs für API Gateway-Ressourcen ist nützlich bei Fehlerbehebung im Betrieb sowie Anwendungsfällen wie Audit und Compliance.

AWS Config kann folgende Änderungen nachverfolgen:

- API-Stufenkonfiguration, wie z. B.:
  - Cache-Cluster-Einstellungen
  - Drosselungseinstellungen
  - Zugriffsprotokolleinstellungen
  - aktive Bereitstellung, die für die Stufe festgelegt wurde
- API-Konfiguration, wie z. B.:
  - Endpunktkonfiguration
  - Version
  - Protokoll
  - Tags

Darüber hinaus können Sie mit der AWS-Config-Regeln-Funktion Konfigurationsregeln definieren und Verstöße gegen diese Regeln automatisch erkennen, nachverfolgen und melden. Indem Sie die Änderungen an diesen Ressourcen-Konfigurationseigenschaften nachverfolgen, können Sie auch durch Änderungen ausgelöste AWS Config-Regeln für Ihre API-Gateway-Ressourcen erstellen und Ihre Ressourcen-Konfigurationen für bewährte Methoden testen.

Sie können AWS Config in Ihrem Konto mithilfe der AWS Config-Konsole oder der AWS CLI aktivieren. Wählen Sie die Ressourcentypen aus, für die Sie Änderungen nachverfolgen möchten. Wenn Sie AWS Config zuvor so konfiguriert haben, dass alle Ressourcentypen aufgezeichnet werden, dann werden diese API-Gateway-Ressourcen automatisch in Ihrem Konto aufgezeichnet. Die Unterstützung für Amazon API Gateway in AWS Config ist in allen öffentlichen AWS-Regionen und in AWS GovCloud (US) verfügbar. Die vollständige Liste der unterstützten Regionen finden Sie unter Amazon-API-Gateway-Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Themen

Unterstützte Ressourcentypen

Die folgenden API-Gateway-Ressourcentypen sind in AWS Config integriert und im Abschnitt Von AWS Config unterstützte AWS-Ressourcentypen und -beziehungen dokumentiert:

- `AWS::ApiGatewayV2::Api` (WebSocket- und HTTP-API)
- `AWS::ApiGateway::RestApi` (REST-API)
- `AWS::ApiGatewayV2::Stage` (WebSocket- und HTTP-API-Stufe)
- `AWS::ApiGateway::Stage` (REST-API-Stufe)

Weitere Informationen über AWS Config finden Sie im AWS Config-Entwicklerhandbuch. Preise finden Sie auf der Seite mit Preisinformationen zu AWS Config.

> **Important**
>
> Wenn Sie eine der folgenden API-Eigenschaften nach Bereitstellung der API ändern, müssen Sie die API erneut bereitstellen, um die Änderungen zu übernehmen. Andernfalls sehen Sie zwar die Attributänderungen in der AWS Config-Konsole, die vorherigen Eigenschaftseinstellungen sind jedoch weiterhin wirksam. Das Laufzeitverhalten der API ist unverändert.
>
> - **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
> - **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

Einrichten von AWS Config

Die Ersteinrichtung von AWS Config wird in den folgenden Themen im AWS Config-Entwicklerhandbuch beschrieben.

- Einrichten von AWS Config mit der Konsole
- Einrichten von AWS Config mit der AWS CLI

Konfigurieren von AWS Config zur Aufzeichnung von API-Gateway-Ressourcen

Standardmäßig zeichnet AWS Config die Konfigurationsänderungen für alle unterstützten Typen von regionalen Ressourcen auf, die in der Region erkannt werden, in der die Umgebung ausgeführt wird. Sie können AWS Config so anpassen, dass Änderungen nur für bestimmte Ressourcentypen oder nur Änderungen an globalen Ressourcen aufgezeichnet werden.

Informationen über regionale im Unterschied zu globalen Ressourcen sowie zum Anpassen der AWS Config-Konfiguration finden Sie unter Auswählen der Ressourcen, die AWS Config aufzeichnet.

Anzeigen von API-Gateway-Konfigurationsdetails in der AWS Config-Konsole

Sie können die AWS Config-Konsole verwenden, um nach API-Gateway-Ressourcen zu suchen und aktuelle und historische Details über deren Konfigurationen zu erhalten. Die folgende Prozedur zeigt, wie Sie Informationen über eine API Gateway-API finden.

So finden Sie eine API-Gateway-Ressource in der AWS-Konfigurationskonsole

1. Öffnen Sie die AWS Config-Konsole.
2. Wählen Sie Resources aus.
3. Wählen Sie auf der Seite Resource inventory (Ressourcenbestand) die Option Resources (Ressourcen) aus.
4. Öffnen Sie das Menü Resource type (Ressourcentyp), blättern Sie zu APIGateway oder APIGatewayV2 und wählen Sie dann einen oder mehrere der API Gateway-Ressourcentypen.
5. Wählen Sie Look up.
6. Wählen Sie in der Liste der Ressourcen, die von AWS Config angezeigt wird, eine Ressource-ID aus. AWS Config zeigt Konfigurationsdetails und andere Informationen über die Ressource, die Sie ausgewählt haben.
7. Die vollständigen Details der aufgezeichneten Konfiguration können Sie mit View Details (Details anzeigen) anzeigen.

Weitere Möglichkeiten zum Suchen einer Ressource und zum Anzeigen von Informationen auf dieser Seite finden Sie unter Anzeigen von AWS-Ressourcenkonfigurationen und -verlauf im AWS Config-Entwicklerhandbuch.

Bewertung von API-Gateway-Ressourcen mithilfe von AWS Config-Regeln

Sie können AWS Config-Regeln erstellen, die die idealen Konfigurationseinstellungen für Ihre API-Gateway-Ressourcen darstellen. Sie können vordefinierte Verwaltete AWS Config-Regeln verwenden oder benutzerdefinierte Regeln definieren. AWS Config verfolgt kontinuierlich Änderungen an der Konfiguration Ihrer Ressourcen, um zu bestimmen, ob diese Änderungen gegen eine Bedingung Ihrer Regeln verstoßen. Die AWS Config-Konsole zeigt den Compliance-Status Ihrer Regeln und Ressourcen an.

Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet wird, kann AWS Config Sie mithilfe eines Amazon-SNS-Themas Entwicklerhandbuch zu Amazon Simple Notification Service warnen. Um die Daten in diesen AWS ConfigWarnungen programmgesteuert zu nutzen, verwenden Sie eine Amazon-SQS-Warteschlange (Amazon Simple Queue Service) als Benachrichtigungsendpunkt für das Amazon-SNS-Thema.

Weitere Informationen zum Einrichten und Verwenden von Regeln finden Sie unter Ressourcen mit Regeln auswerten im AWS Config-Entwicklerhandbuch.

Compliance-Validierung für Amazon API Gateway

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.

- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen erstellen AWS können.

  > **Note**
  >
  > AWS-Services Nicht alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Zuverlässigkeit in Amazon API Gateway

Im Zentrum der globalen AWS Infrastruktur stehen die AWS-Regionen und Availability Zones (Verfügbarkeitszonen, AZs). AWS -Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über hoch redundante Netzwerke mit niedriger Latenz und hohen Durchsätzen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen über AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Um zu verhindern, dass Ihre APIs von zu vielen Anfragen überlastet werden, drosselt API Gateway Anfragen an Ihre APIs. Insbesondere setzt API Gateway ein Limit für eine Steady-State-Rate und einen Burst von Anfragen an alle APIs in Ihrem Konto. Sie können die benutzerdefinierte Drosselung für Ihre APIs konfigurieren. Weitere Informationen hierzu finden Sie unter Drosseln der API-Anforderungen für einen besseren Durchsatz.

Sie können Route-53-Zustandsprüfungen verwenden, um das DNS-Failover von einer API-Gateway-API in einer primären Region zu einer API-Gateway-API in einer sekundären Region zu kontrollieren. Ein Beispiel finden Sie unter the section called “DNS-Failover”.

Infrastruktursicherheit in Amazon API Gateway

Als verwalteter Service ist Amazon API Gateway durch die globalen Verfahren zur Gewährleistung der Netzwerksicherheit von AWS geschützt. Informationen zu AWS-Sicherheitsdiensten und wie AWS die Infrastruktur schützt, finden Sie unter AWSCloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS-Umgebung anhand der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz im Security Pillar AWS Well-Architected Framework.

Sie verwenden von AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf API Gateway zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Diese API-Operationen lassen sich von einem beliebigen Netzwerkstandort aus aufrufen. Da API Gateway jedoch ressourcenbasierte Zugriffsrichtlinien unterstützt, kann es zu Einschränkungen bezüglich der Quell-IP-Adresse kommen. Sie können auch ressourcenbasierte Richtlinien verwenden, um den Zugriff über bestimmte Amazon Virtual Private Cloud (Amazon VPC)-Endpunkte oder bestimmte VPCs zu steuern. Dadurch wird der Netzwerkzugriff auf eine bestimmte API-Gateway-Ressource effektiv auf die spezifische VPC innerhalb des AWS-Netzwerks isoliert.

Schwachstellenanalyse in Amazon API Gateway

Konfiguration und IT-Steuerung unterliegen der übergreifenden Verantwortlichkeit von AWS und Ihnen, unserem Kunden. Weitere Informationen finden Sie unter AWSModell der übergreifenden Verantwortlichkeit.

Bewährte Methoden in Amazon API Gateway

API Gateway bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Implementieren der geringstmöglichen Zugriffsrechte

Verwenden Sie IAM-Richtlinien zur Implementierung des Zugriffs mit den geringsten Rechten zum Erstellen, Lesen, Aktualisieren oder Löschen von API Gateway-APIs. Weitere Informationen hierzu finden Sie unter Identitäts- und Zugriffsverwaltung für Amazon API Gateway. API Gateway bietet mehrere Optionen zur Steuerung des Zugriffs auf von Ihnen erstellte APIs. Weitere Informationen hierzu finden Sie unter Zugriff auf eine REST-API in API Gateway steuern und verwalten, Steuern und Verwalten des Zugriffs auf eine WebSocket API in API Gateway und Steuern des Zugriffs auf HTTP-APIs mit JWT-Genehmigern.

Implementieren der Protokollierung

Verwenden Sie CloudWatch Protokolle oder Amazon Data Firehose, um Anfragen an Ihre APIs zu protokollieren. Weitere Informationen hierzu finden Sie unter Überwachung von REST-APIs, Protokollierung für eine WebSocket API konfigurieren und Protokollierung für eine HTTP-API konfigurieren.

Amazon- CloudWatch Alarme implementieren

Mithilfe von CloudWatch Alarmen überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird eine Benachrichtigung an ein Amazon Simple Notification Service-Thema oder eine AWS Auto Scaling Richtlinie gesendet. CloudWatch Alarme rufen keine Aktionen auf, wenn sich eine Metrik in einem bestimmten Status befindet. Der Status muss sich stattdessen geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein. Weitere Informationen finden Sie unter the section called “CloudWatch Metriken”.

Aktivieren von AWS CloudTrail

CloudTrail bietet eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem - AWS Service in API Gateway durchgeführten Aktionen. Anhand der von CloudTrailgesammelten Informationen können Sie die an API Gateway gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen. Weitere Informationen finden Sie unter the section called “Arbeiten mit CloudTrail”.

Aktivieren von AWS Config

AWS Config bietet eine detaillierte Ansicht der Konfiguration der AWS Ressourcen in Ihrem Konto. Sie können Ressourcenbeziehungen betrachten, einen Verlauf der Konfigurationsänderungen anzeigen und feststellen, wie sich Beziehungen und Konfigurationen im Lauf der Zeit ändern. Sie können verwenden, AWS Config um Regeln zu definieren, die Ressourcenkonfigurationen auf Datenkonformität auswerten. - AWS Config Regeln stellen die idealen Konfigurationseinstellungen für Ihre API Gateway-Ressourcen dar. Wenn eine Ressource gegen eine Regel verstößt und als nicht konform gekennzeichnet wird, AWS Config kann Sie mithilfe eines Amazon Simple Notification Service (Amazon SNS)-Themas benachrichtigen. Details hierzu finden Sie unter the section called “Arbeiten mit AWS Config”.

Verwenden von AWS Security Hub

Überwachen Sie Ihre API-Gateway-Nutzung in Bezug auf bewährte Sicherheitsmethoden, indem Sie AWS Security Hub verwenden. Security Hub verwendet Sicherheitskontrollen für die

Bewertung von Ressourcenkonfigurationen und Sicherheitsstandards, um Sie bei der Einhaltung verschiedener Compliance-Frameworks zu unterstützen. Weitere Informationen zur Verwendung von Security Hub für die Bewertung von API-Gateway-Ressourcen finden Sie unter Amazon-API-Gateway-Steuerelemente im AWS Security Hub -Benutzerhandbuch.

API Gateway-Ressourcen taggen

Ein Tag ist ein Metadaten-Label, das Sie zuweisen oder das einer AWS Ressource AWS zugewiesen wird. Jedes Tag besteht aus zwei Teilen:

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment` oder `Project`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- einem optionalen Feld, dem sogenannten Tag-Wert (z. B. `111122223333` oder `Production`). Ein nicht angegebener Tag-Wert entspricht einer leeren Zeichenfolge. Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß-/Kleinschreibung unterschieden.

Tags sind für folgende Aktivitäten nützlich:

- Steuern des Zugriffs auf Ihre Ressourcen basierend auf den ihnen zugeordneten Tags. Sie können den Zugriff steuern, indem Sie in den Bedingungen für eine AWS Identity and Access Management -Richtlinie (IAM) Tag-Schlüssel und -Werte angeben. Weitere Informationen zur Tag-basierten Zugriffskontrolle finden Sie unter Steuern des Zugriffs mit Tags im IAM-Benutzerhandbuch.
- Verfolgen Sie Ihre AWS Kosten. Sie aktivieren diese Tags auf dem AWS Billing and Cost Management Dashboard. AWS verwendet die Tags, um Ihre Kosten zu kategorisieren und Ihnen einen monatlichen Kostenverteilungsbericht zu senden. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im AWS Billing -Benutzerhandbuch.
- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS Dienste unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verknüpft sind. Sie könnten beispielsweise einer API-Gateway-Stufe dasselbe Tag zuweisen, das Sie einer CloudWatch Event-Regel zuweisen.

Tipps zur Verwendung von Tags finden Sie im Whitepaper AWS Tagging-Strategien.

Die folgenden Abschnitte enthalten weitere Informationen über Tags für Amazon API Gateway.

Themen

API Gateway-Ressourcen, die mit Tags versehen werden können

Tags können für die folgenden HTTP-API- oder WebSocket API-Ressourcen in der Amazon API Gateway V2 API festgelegt werden:

- `Api`
- `DomainName`
- `Stage`
- `VpcLink`

Darüber hinaus können Tags für die folgenden REST-API-Ressourcen in der Amazon API Gateway V1-API festgelegt werden:

- `ApiKey`
- `ClientCertificate`
- `DomainName`
- `RestApi`
- `Stage`
- `UsagePlan`
- `VpcLink`

Tags können nicht direkt in anderen Ressourcen festgelegt werden. In der Amazon API Gateway V1-API erben untergeordnete Ressourcen die Tags, die für übergeordneten Ressourcen festgelegt sind. Beispielsweise:

- Wenn ein Tag in einer `RestApi`-Ressource festgelegt ist, wird dieses Tag an die folgenden untergeordneten Ressourcen dieser `RestApi` für Attributbasierte Zugriffskontrolle vererbt:
  - `Authorizer`
  - `Deployment`
  - `Documentation`
  - `GatewayResponse`
  - `Integration`
  - `Method`

- `Model`
- `Resource`
- `ResourcePolicy`
- `Setting`
- `Stage`

- Wenn ein Tag in einem `DomainName` festgelegt ist, wird dieses Tag von allen `BasePathMapping`-Ressourcen darunter geerbt.
- Wenn ein Tag in einem `UsagePlan` festgelegt ist, wird dieses Tag von allen `UsagePlanKey`-Ressourcen darunter geerbt.

> **Note**
>
> Die Tag-Vererbung gilt nur für attributbasierte Zugriffskontrolle. Sie können beispielsweise keine vererbten Tags verwenden, um die Kosten in zu überwachen AWS Cost Explorer. API Gateway gibt keine geerbten Tags zurück, wenn Sie GetTagseine Ressource aufrufen.

Tag-Vererbung in der Amazon API Gateway V1-API

Zuvor war es nur möglich, Tags in Stufen festzulegen. Da Sie sie jetzt auch in anderen Ressourcen festlegen können, kann eine `Stage` ein Tag auf zwei Arten erhalten:

- Das Tag kann direkt in der festgelegt werde `Stage`.
- Die Stufe kann das Tag von der übergeordneten erbe `RestApi`.

Wenn eine Stufe ein Tag auf beide Arten erhält, hat das Tag den Vorrang, das direkt in der Stufe festgelegt wurde. Nehmen wir beispielsweise an, eine Stufe erbt die folgenden Tags von der übergeordneten REST-API:

```
{
 'foo': 'bar',
 'x':'y'
}
```

Nehmen wir an, es werden auch die folgenden Tags direkt in ihr festgelegt:

```
{
 'foo': 'bar2',
 'hello': 'world'
}
```

Die Wirkung für die Stufe wäre, dass sie über folgende Tags mit den folgenden Werten verfügt:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x':'y'
}
```

Tag-Einschränkungen und Nutzungskonventionen

Die folgenden Einschränkungen und Nutzungskonventionen gelten für die Verwendung von Tags mit API Gateway-Ressourcen:

- Jede Ressource kann maximal 50 Tags haben.
- Jeder Tag muss für jede Ressource eindeutig sein. Jeder Tag kann nur einen Wert haben.
- Die maximale Länge des Tag-Schlüssels beträgt 128 Unicode-Zeichen in UTF-8.
- Die maximale Länge des Tag-Wertes beträgt 256 Unicode-Zeichen in UTF-8.
- Zulässige Zeichen für Schlüssel und Werte sind Buchstaben, Zahlen, Leerzeichen, die in UTF-8 darstellbar sind, und die folgenden Zeichen: . : + = @ _ / - (Bindestrich). Für Amazon EC2-Ressourcen können beliebige Zeichen verwendet werden.
- Bei Tag-Schlüsseln und -Werten muss die Groß-/Kleinschreibung beachtet werden. Eine bewährte Methode besteht darin, sich für eine einheitliche Schreibweise der Tag-Benennungen zu entscheiden und diese Strategie für alle Ressourcentypen umzusetzen. Entscheiden Sie sich beispielsweise für `Costcenter`, `costcenter` oder `CostCenter` und verwenden Sie diese Konvention für alle Tags. Vermeiden Sie die Verwendung von ähnlichen Tags mit uneinheitlicher Fallunterscheidung.
- Das Präfix `aws:` darf nicht für Tags verwendet werden; es ist für die AWS -Verwendung reserviert. Sie können keine Tag-Schlüssel oder -Werte mit diesem Präfix bearbeiten oder löschen. Tags mit diesem Präfix werden nicht als Ihre Tags pro Ressourcenlimit angerechnet.

Tags zur Steuerung des Zugriffs auf API Gateway-REST-API-Ressourcen verwenden

Bedingungen in AWS Identity and Access Management Richtlinien sind Teil der Syntax, mit der Sie Berechtigungen für API-Gateway-Ressourcen angeben. Einzelheiten zur Angabe von IAM-Richtlinien finden Sie unter the section called “Verwenden von IAM-Berechtigungen”. In API Gateway können Ressourcen Tags haben. Einige Aktionen können ebenfalls Tags enthalten. Wenn Sie eine IAM-Richtlinie erstellen, können Sie Tag-Bedingungsschlüssel verwenden, um Folgendes zu kontrollieren:

- Welche Benutzer auf der Grundlage von Tags, die die Ressource bereits besitzt, Aktionen mit einer API Gateway-Ressource ausführen können.
- Welche Tags in der Anforderung einer Aktion übergeben werden können.
- Ob bestimmte Tag-Schlüssel in einer Anforderung verwendet werden können.

Die Verwendung von Tags für die attributbasierte Zugriffskontrolle kann eine feinere Kontrolle als die Kontrolle auf API-Ebene sowie eine dynamischere Kontrolle als die ressourcenbasierte Zugriffskontrolle ermöglichen. IAM-Richtlinien können erstellt werden, die eine Operation basierend auf Tags in der Anforderung (Anforderungs-Tags) oder Tags in der Ressource, mit der gearbeitet wird (Ressourcen-Tags), zulassen oder ablehnen. Im Allgemeinen sind Ressourcen-Tags für Ressourcen vorgesehen, die bereits vorhanden ist. Anforderungs-Tags sind dafür da, wenn Sie neue Ressourcen erstellen.

Die vollständige Syntax und Semantik der Tag-Bedingungsschlüssel finden Sie unter Steuern des Zugriffs mit Tags im IAM-Benutzerhandbuch.

Die folgenden Beispiele zeigen, wie Tag-Bedingungen in Richtlinien für API Gateway-Benutzer angegeben werden können.

Einschränken von Aktionen basierend auf Ressourcen-Tags

Die folgende Beispielrichtlinie erteilt Benutzern die Berechtigung zum Ausführen aller Aktionen für alle Ressourcen, sofern diese Ressourcen nicht über das Tag „`Environment`“ mit dem Wert „`prod`“ verfügen.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "apigateway:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}
```

Zulassen von Aktionen basierend auf Ressourcen-Tags

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle Aktionen für API Gateway-Ressourcen auszuführen, sofern die Ressourcen über das Tag „`Environment`“ mit dem Wert „`Development`“ verfügen. Die `Deny`-Anweisung verhindert, dass der Benutzer den Wert des `Environment`-Tags ändert.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConditionallyAllow",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "Development"
        }
```

```
      }
    },
    {
      "Sid": "AllowTagging",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ]
    },
    {
      "Sid": "DenyChangingTag",
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ],
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "Environment"
        }
      }
    }
  ]
}
```

Tagging-Operationen verweigern

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle API Gateway-Aktionen auszuführen, mit Ausnahme des Änderns von Tags.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
```

```
      "Resource": [
        "*"
      ],
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "arn:aws:apigateway:*::/tags*",
    }
  ]
}
```

Tagging-Operationen erlauben

Die folgende Beispielrichtlinie erlaubt es Benutzern, alle API Gateway-Ressourcen abzurufen und die Tags für diese Ressourcen zu ändern. Um die Tags für eine Ressource abzurufen, muss der Benutzer über `GET`-Berechtigungen für diese Ressource verfügen. Um die Tags für eine Ressource zu aktualisieren, muss der Benutzer über `PATCH`-Berechtigungen für diese Ressource verfügen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*",
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PATCH",
      ],
      "Resource": [
```

```
        "arn:aws:apigateway:*::*",
      ]
    }
  ]
}
```

API-Referenzen

Amazon API Gateway bietet APIs für die Erstellung und Bereitstellung Ihres eigenen HTTP-Codes und Ihrer eigenen WebSocket APIs. Darüber hinaus sind API Gateway Gateway-APIs in AWS Standard-SDKs verfügbar.

Wenn Sie eine Sprache verwenden, für die ein AWS SDK existiert, ziehen Sie es möglicherweise vor, das SDK zu verwenden, anstatt die API-Gateway-REST-APIs direkt zu verwenden. Die SDKs vereinfachen die Authentifizierung, lassen sich leicht in die Entwicklungsumgebung integrieren und bieten einen einfachen Zugriff auf die API Gateway-Befehle.

Hier finden Sie die AWS SDKs und die API Gateway REST API-Referenzdokumentation:

- Tools für Amazon Web Services
- REST-API-Referenz für Amazon API Gateway
- Amazon API Gateway WebSocket und HTTP-API-Referenz

Amazon API Gateway-Kontingente und wichtige Hinweise

Themen

Sofern nicht anders angegeben, können die Kontingente auf Anfrage erhöht werden. Um eine Kontingenterhöhung zu beantragen, können Sie Service Quotas verwenden oder sich an das AWS Support Center wenden.

Wenn für eine Methode die Autorisierung aktiviert ist, beträgt die maximale Länge des ARN der Methode (z. B. `arn:aws:execute-api:{region-id}:{account-id}:{api-id}/{stage-id}/{method}/{resource}/{path}`) 1600 Byte. Die Pfadparameterwerte (deren Größe zur Laufzeit bestimmt wird) können dazu führen, dass die ARN-Länge den Grenzwert überschreitet. In diesem Fall erhält der API-Client eine `414 Request URI too long`-Antwort.

> **Note**
>
> Hierdurch wird die Länge des URI begrenzt, wenn Ressourcenrichtlinien verwendet werden. Im Fall privater APIs, für die eine Ressourcenrichtlinie erforderlich ist, wird hierdurch die URI-Länge aller privaten APIs begrenzt.

API Gateway-Kontingent auf Kontoebene, pro Region

Die folgenden Kontingente gelten pro Konto und pro Region in Amazon API Gateway.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Drosseln Sie das Kontingent pro Konto, pro Region für HTTP-APIs, WebSocket REST-APIs, APIs und WebSocket Callback-APIs	10 000 Anforderungen pro Sekunde (RPS) mit einer zusätzlichen Steigerungskapazität über den Token-Bucket-Algorithmus, unter Verwendung einer maximalen Bucket-Kapazität von 5 000 Anforderungen. * **Note** Das Burst-Kontingent wird vom API Gateway Service-Team auf der Grundlage des Gesamt-RPS-Kontingents für das Konto in der Region festgelegt. Kunden können dieses Kontingent nicht kontrollieren und keine Änderungen daran anfordern.	Ja
Regionale APIs	600	Nein
Edge-optimierte APIs	120	Nein

* Für die folgenden Regionen beträgt das Standard-Drosselungskontingent 2500 RPS und das Standard-Burst-Kontingent 1250 RPS: Afrika (Kapstadt), Europa (Mailand), Asien-Pazifik (Jakarta), Naher Osten (VAE), Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Europa (Spanien), Europa (Zürich), Israel (Tel Aviv) und Kanada West (Calgary).

HTTP-API-Kontingente

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer HTTP-API in API Gateway.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Routen pro API	300	Ja
Integrationen pro API	300	Nein
Maximales Timeout für Integrationen	30 Sekunden	Nein
Stufen pro API	10	Ja
API-Zuordnungen auf mehreren Ebenen pro Domain	200	Nein
Tags pro Stufe	50	Nein
Gesamte kombinierte Größe der Anfragezeilen- und Header-Werte	10240 Byte	Nein
Nutzlastgröße	10 MB	Nein
Benutzerdefinierte Domänen pro Konto pro Region	120	Ja
Größe der Zugriffsprotokollvorlage	3 KB	Nein
Amazon CloudWatch Logs-Protokolleintrag	1 MB	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Genehmiger pro API	10	Ja
Zielgruppen pro Genehmiger	50	Nein
Bereiche pro Route	10	Nein
Timeout für den JSON Web Key Set-Endpunkt	1.500 ms	Nein
Antwortgröße des JSON Web Key Set-Endpunkts	150000 Bytes	Nein
Timeout für OpenID Connect-Erkennungsendpunkt	1.500 ms	Nein
Timeout der Antwort des Lambda-Genehmigers	10000 ms	Nein
VPC-Links pro Konto pro Region	10	Ja
Subnetze pro VPC-Link	10	Ja
Stufenvariablen pro Stufe	100	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Länge (in Zeichen) des Schlüssels in einer Stufenvariable	64	Nein
Länge (in Zeichen) des Werts in einer Stufenvariable	512	Nein

API Gateway Gateway-Kontingente für die Konfiguration und Ausführung einer WebSocket API

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer WebSocket API in Amazon API Gateway.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Neue Verbindungen pro Sekunde pro Konto (über alle WebSocket APIs hinweg) pro Region	500	Ja
Gleichzeitige Verbindungen	Nicht zutreffend *	Nicht zutreffend

Ressource oder Operation	Standardkontingent	Kann erhöht werden
AWS Lambda Autorisierer pro API	10	Ja
AWS Lambda Größe des Autorisierer-Ergebnisses	8 KB	Nein
Routen pro API	300	Ja
Integrationen pro API	300	Ja
Integrations-Timeout	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Nein
Stufen pro API	10	Ja
WebSocket Größe des Frames	32 KB	Nein
Nachrichten-Nutzlastgröße	128 KB *	Nein
Verbindungsdauer für die WebSocket API	2 Stunden	Nein
Zeitlimit für Verbindungsleerlauf	10 Minuten	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Länge der URL für eine WebSocket API in Zeichen	4096	Nein

* API Gateway erzwingt kein Kontingent für gleichzeitige Verbindungen. Die maximale Anzahl gleichzeitiger Verbindungen wird durch die Rate neuer Verbindungen pro Sekunde und die maximale Verbindungsdauer von zwei Stunden bestimmt. Wenn Clients beispielsweise mit dem Standardkontingent von 500 neuen Verbindungen pro Sekunde eine Verbindung mit der maximalen Rate über zwei Stunden herstellen, kann API Gateway bis zu 3 600 000 gleichzeitige Verbindungen bedienen.

** Aufgrund des Kontingents für die WebSocket Framegröße von 32 KB muss eine Nachricht, die größer als 32 KB ist, in mehrere Frames aufgeteilt werden, die jeweils 32 KB oder kleiner sind. Dies gilt für `@connections`-Befehle. Wenn eine größere Nachricht (oder eine größere Frame-Größe) empfangen wird, wird die Verbindung mit dem Code 1009 geschlossen.

API Gateway-Kontingente für die Konfiguration und Ausführung einer REST-API

Die folgenden Kontingente gelten für die Konfiguration und Ausführung einer REST-API in Amazon API Gateway. Für restapi:import oder restapi:put beträgt die maximale Größe der API-Definitionsdatei 6 MB.

Alle Kontingente, die pro API gelten, können nur für spezifische APIs erhöht werden.

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Benutzerdefinierte Domänennamen	120	Ja

Ressource oder Operation	Standardkontingent	Kann erhöht werden
pro Konto pro Region		
API-Zuordnungen auf mehreren Ebenen pro Domain	200	Nein
Länge (in Zeichen) der URL für eine Edge-optimierte API	8192	Nein
Länge (in Zeichen) der URL für eine regionale API	10240	Nein
Private APIs pro Konto pro Region	600	Nein
Länge in Zeichen der API Gateway-Ressourcenrichtlinie	8192	Ja
API-Schlüssel pro Konto pro Region	10000	Nein
Clientzertifikate pro Konto pro Region	60	Ja
Autorisierer pro API (AWS Lambda und Amazon Cognito)	10	Ja

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Dokumentationsteile pro API	2000	Ja
Ressourcen pro API	300	Ja
Stufen pro API	10	Ja
Stufenvariablen pro Stufe	100	Nein
Länge (in Zeichen) des Schlüssels in einer Stufenvariable	64	Nein
Länge (in Zeichen) des Werts in einer Stufenvariable	512	Nein
Nutzungspläne pro Konto pro Region	300	Ja
Nutzungspläne pro API-Schlüssel	10	Ja
VPC-Links pro Konto pro Region	20	Ja
API-Caching-TTL	300 Sekunden standardmäßig und konfigurierbar zwischen 0 und 3.600 durch einen API-Eigentümer.	Nicht für die obere Grenze (3.600)

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Zwischeng espeicherte Antwortgröße	1048576 Bytes Durch Datenverschlüsselung kann sich die Größe des zwischengespeicherten Elements erhöhen.	Nein
Integrations-Timeo ut	50 Millisekunden — 29 Sekunden für alle Integrationstypen, einschließlich Lambda, Lambda-Proxy, HTTP, HTTP-Proxy und Integrationen. AWS	Ja*
Gesamte kombinier te Größe aller Header-Werte	10 240 Bytes	Nein
Gesamte kombinier te Größe aller Header-Werte für eine private API	8 000 Byte	Nein
Nutzlastgröße	10 MB	Nein
Tags pro Stufe	50	Nein
Anzahl der Iterationen in einer `#foreach ...` `#end`-Schleife in Zuordnung svorlagen	1000	Nein
ARN-Länge einer Methode mit Autorisierung	1600 Bytes	Nein

Ressource oder Operation	Standardkontingent	Kann erhöht werden
Ablehnung seinstellungen auf Methodenebene für eine Phase in einem Nutzungsplan.	20	Ja
Modellgröße pro API	400 KB	Nein
Anzahl der Zertifikate in einem Truststore	1.000 Zertifikate mit einer Gesamtobjektgröße von bis zu 1 MB.	Nein

* Sie können das Integrations-Timeout nicht auf weniger als 50 Millisekunden festlegen. Sie können das Integrations-Timeout für regionale APIs und private APIs auf mehr als 29 Sekunden erhöhen. Dies kann jedoch eine Reduzierung Ihrer Drosselungsquote auf Kontoebene erfordern.

API Gateway-Kontingente für das Erstellen, Bereitstellen und Verwalten einer API

Die folgenden festen Kontingente gelten für die Erstellung, Bereitstellung und Verwaltung einer API in API Gateway mithilfe der AWS CLI API Gateway Gateway-Konsole oder der API Gateway Gateway-REST-API und ihrer SDKs. Diese Kontingente können nicht erhöht werden.

Action	Standardkontingent	Kann erhöht werden
CreateApiKey	5 Anforderungen pro Sekunde pro Konto	Nein
CreateDeployment	1 Anforderung alle 5 Sekunden pro Konto	Nein

Action	Standardkontingent	Kann erhöht werden
CreateDocumentationVersion	1 Anforderung alle 20 Sekunden pro Konto	Nein
CreateDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
CreateResource	5 Anforderungen pro Sekunde pro Konto	Nein
CreateRestApi	Regionale oder private API • 1 Anforderung alle 3 Sekunden pro Konto Edge-optimierte API • 1 Anforderung alle 30 Sekunden pro Konto	Nein
CreateVpcLink(V2)	1 Anforderung alle 15 Sekunden pro Konto	Nein
DeleteApiKey	5 Anforderungen pro Sekunde pro Konto	Nein
DeleteDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
DeleteResource	5 Anforderungen pro Sekunde pro Konto	Nein
DeleteRestApi	1 Anforderung alle 30 Sekunden pro Konto	Nein
GetResources	5 Anforderungen alle 2 Sekunden pro Konto	Nein

Action	Standardkontingent	Kann erhöht werden
DeleteVpcLink(V2)	1 Anforderung alle 30 Sekunden pro Konto	Nein
ImportDocumentationParts	1 Anforderung alle 30 Sekunden pro Konto	Nein
ImportRestApi	Regionale oder private API • 1 Anforderung alle 3 Sekunden pro Konto Edge-optimierte API • 1 Anforderung alle 30 Sekunden pro Konto	Nein
PutRestApi	1 Anforderung pro Sekunde pro Konto	Nein
UpdateAccount	1 Anforderung alle 20 Sekunden pro Konto	Nein
UpdateDomainName	1 Anforderung alle 30 Sekunden pro Konto	Nein
UpdateUsagePlan	1 Anforderung alle 20 Sekunden pro Konto	Nein
Andere Operationen	Kein Kontingent bis zum Gesamtkontokontingent.	Nein
Gesamte Operationen	10 Anforderungen pro Sekunde mit einem Burst-Kontingent von 40 Anforderungen pro Sekunde.	Nein

Wichtige Hinweise zu Amazon API Gateway

Themen

Wichtige Hinweise zu Amazon API Gateway für REST-APIs, HTTP-APIs und WebSocket APIs

- Signature Version 4A wird von Amazon API Gateway nicht offiziell unterstützt.

Amazon API Gateway — wichtige Hinweise zu REST und WebSocket APIs

- API Gateway unterstützt nicht die gemeinsame Nutzung eines benutzerdefinierten Domainnamens für REST und WebSocket APIs.
- Phasennamen dürfen nur alphanumerische Zeichen sowie Binde- und Unterstriche enthalten. Die maximale Länge beträgt 128 Zeichen.
- Die Pfade `/ping` und `/sping` sind für die Servicezustandsprüfung reserviert. Wenn diese für API-Ressourcen auf Stammebene mit benutzerdefinierten Domains verwendet werden, liefern sie nicht das erwartete Ergebnis.
- API Gateway begrenzt Protokollereignisse derzeit auf 1024 Bytes. Protokollereignisse, die größer als 1024 Byte sind, wie Anforderungs- und Antworttexte, werden von API Gateway vor der Übermittlung an CloudWatch Logs gekürzt.
- CloudWatch Metrics begrenzt derzeit Dimensionsnamen und -werte auf 255 gültige XML-Zeichen. (Weitere Informationen finden Sie im CloudWatch Benutzerhandbuch.) Dimensionenwerte sind eine Funktion benutzerdefinierter Namen, einschließlich API-Name, Bezeichnung (Stufenname) und Ressourcenname. Achten Sie bei der Auswahl dieser Namen darauf, die CloudWatch Metrik-Grenzwerte nicht zu überschreiten.
- Die maximale Größe einer Zuordnungsvorlage beträgt 300 KB.

Amazon API Gateway — wichtige Hinweise für WebSocket APIs

- API Gateway unterstützt Nachrichten-Payloads bis zu 128 KB mit einer maximalen Frame-Größe von 32 KB. Sie müssen Nachrichten, die 32 KB überschreiten, in mehrere Frames aufteilen, die jeweils 32 KB oder kleiner sind. Wenn eine größere Nachricht empfangen wird, wird die Verbindung mit Code 1009 geschlossen.

Amazon API Gateway – Wichtige Hinweise für REST-APIs

- Das Pipe-Klartextzeichen (`|`) wird in Abfragezeichenfolgen zur Anforderung einer URL nicht unterstützt und muss URL-kodiert werden.
- Das Semikolon (`;`) wird in Abfragezeichenfolgen zur Anforderung einer URL nicht unterstützt und führt dazu, dass die Daten aufgeteilt werden.
- REST-APIs dekodieren URL-kodierte Anforderungsparameter, bevor sie an Backend-Integrationen übergeben werden. Bei UTF-8-Anforderungsparametern dekodieren REST-APIs die Parameter und übergeben sie dann als Unicode an Backend-Integrationen.
- Beim Verwenden der Amazon API Gateway-Konsole zum Testen einer API können Sie die Antwort "unknown endpoint errors" erhalten, wenn auf dem Backend ein selbstsigniertes Zertifikat präsentiert wird, in der Zertifikatkette ein Zwischenzertifikat fehlt oder das Backend eine andere, nicht erkennbare zertifikatbezogenen Ausnahme auslöst.
- Sie sollten eine API-`Resource` oder `Method`-Entität mit einer privaten Integration löschen, nachdem Sie hartcodierte Referenzen eines `VpcLink` entfernt haben. Andernfalls ist die Integration nicht abgeschlossen und Sie erhalten eine Fehlermeldung mit dem Hinweis, dass der VPC-Link noch verwendet wird, obwohl die Entität `Resource` oder `Method` gelöscht wurde. Dieses Verhalten liegt nicht vor, wenn die private Integration über eine Stufenvariable auf `VpcLink` verweist.
- Die folgenden Backends unterstützen die SSL-Clientauthentifizierung möglicherweise nicht auf eine mit API Gateway kompatible Weise:
  - NGINX
  - Heroku
- API Gateway unterstützt die meisten der OpenAPI-2.0-Spezifikation und der OpenAPI-3.0-Spezifikation mit den folgenden Ausnahmen:
  - Pfadsegmente dürfen nur alphanumerische Zeichen, Unterstriche, Bindestriche, Punkte, Kommas, Doppelpunkte und geschweifte Klammern enthalten. Pfadparameter müssen als

separate Pfadsegmente vorliegen. Beispiel: "resource/{path_parameter_name}" ist gültig, "resource{path_parameter_name}" nicht.

- Modellnamen dürfen nur alphanumerische Zeichen enthalten.
- Als Eingabeparameter werden nur die folgenden Attribute unterstützt: `name`, `in`, `required`, `type`, `description`. Andere Attribute werden ignoriert.
- Der `securitySchemes`-Typ muss bei Verwendung `apiKey` lauten. Die OAuth 2- und HTTP-Standardauthentifizierung wird jedoch über Lambda-Genehmiger unterstützt die OpenAPI-Konfiguration wird über Anbietererweiterungen ermöglicht.
- Das Feld `deprecated` wird nicht unterstützt und aus exportierten APIs entfernt.
- API Gateway-Modelle werden unter Verwendung von JSON-Schemaentwurf 4 anstelle des von OpenAPI verwendeten JSON-Schemas definiert.
- Der Parameter `discriminator` wird in Schemaobjekten nicht unterstützt.
- Das Tag `example` wird nicht unterstützt.
- `exclusiveMinimum` wird von API Gateway nicht unterstützt.
- Die Tags `maxItems` und `minItems` werden bei der einfachen Anforderungsvalidierung nicht berücksichtigt. Um dieses Problem zu umgehen, aktualisieren Sie das Modell nach dem Import, bevor Sie die Validierung vornehmen.
- `oneOf` wird für OpenAPI 2.0 oder die SDK-Generierung nicht unterstützt.
- Das Feld `readOnly` wird nicht unterstützt.
- `$ref` kann nicht für den Verweis auf andere Dateien verwendet werden.
- Antwortdefinitionen des Formulars `"500": {"$ref": "#/responses/UnexpectedError"}` werden im OpenAPI-Basisverzeichnis nicht unterstützt. Ersetzen Sie die Referenz durch das Inline-Schema, um dieses Problem zu umgehen.
- Zahlen vom Typ `Int32` oder `Int64` werden nicht unterstützt. Ein Beispiel sehen Sie unten:

```
"elementId": {
    "description": "Working Element Id",
    "format": "int32",
    "type": "number"
}
```

- Der Formattyp Dezimalzahl (`"format": "decimal"`) wird in Schemadefinitionen nicht unterstützt.

- In Methodenantworten muss die Schemadefinition ein Objekttyp sein und darf keine primitiven Datentypen umfassen. Beispielsweise wird `"schema": { "type": "string"}` nicht unterstützt. Sie können dies jedoch umgehen, indem Sie den folgenden Objekttyp verwenden:

```
"schema": {
     "$ref": "#/definitions/StringResponse"
           }

 "definitions": {
    "StringResponse": {
      "type": "string"
    }
  }
```

  - API Gateway verwendet keine in der OpenAPI-Spezifikation definierte Sicherheit auf Stammebene. Daher muss die Sicherheit auf Vorgangsebene definiert werdne, um korrekt angewendet werden zu können.
  - Das `default`-Schlüsselwort wird nicht unterstützt.
- API Gateway führt die folgenden Beschränkungen und Limitierungen bei der Handhabung von Methoden mit Lambda- oder HTTP-Integration durch.
  - Bei der Verarbeitung von Header-Namen und Abfrageparametern wird die Groß- und Kleinschreibung beachtet.
  - Die folgende Tabelle listet die Header auf, die gelöscht, erneut zugewiesen oder anderweitig modifiziert werden können, wenn sie an den Integrationsendpunkt oder von diesem zurückgesendet werden. In dieser Tabelle:
    - `Remapped` bedeutet, dass der Header-Name von *`<string>`* in `X-Amzn-Remapped-`*`<string>`* geändert wird.

      `Remapped Overwritten` bedeutet, dass der Header-Name von *`<string>`* in `X-Amzn-Remapped-`*`<string>`* geändert und der Wert überschrieben wird.

Header-Name	Anfrage (**http**/**http_proxy** /**lambda**)	Antwort (**http**/**http_proxy** /**lambda**)
`Age`	Pass-Through	Pass-Through
`Accept`	Pass-Through	Gelöscht/Pass-Through/Pass-Through
`Accept-Charset`	Pass-Through	Pass-Through
`Accept-Encoding`	Pass-Through	Pass-Through
`Authorization`	Pass-Through*	Remapped
`Connection`	Pass-Through/Pass-Through/Gelöscht	Remapped
`Content-Encoding`	Pass-Through/Gelöscht/Pass-Through	Pass-Through
`Content-Length`	Pass-Through (generiert auf der Grundlage des Inhalts)	Pass-Through
`Content-MD5`	Gelöscht	Remapped

Header-Name	Anfrage (**http**/**http_proxy** /**lambda**)	Antwort (**http**/**http_proxy** /**lambda**)
`Content-Type`	Pass-Through	Pass-Through
`Date`	Pass-Through	Neu zugeordnet überschrieben
`Expect`	Gelöscht	Gelöscht
`Host`	Auf den Integrationsendpunkt überschrieben	Gelöscht
`Max-Forwards`	Gelöscht	Remapped
`Pragma`	Pass-Through	Pass-Through
`Proxy-Authenticate`	Gelöscht	Gelöscht
`Range`	Pass-Through	Pass-Through
`Referer`	Pass-Through	Pass-Through

Header-Name	Anfrage (**http**/**http_proxy** /**lambda**)	Antwort (**http**/**http_proxy** /**lambda**)
`Server`	Gelöscht	Neu zugeordnet überschrieben
`TE`	Gelöscht	Gelöscht
`Transfer-Encoding`	Gelöscht/Gelöscht/Ausnahme	Gelöscht
`Trailer`	Gelöscht	Gelöscht
`Upgrade`	Gelöscht	Gelöscht
`User-Agent`	Pass-Through	Remapped
`Via`	Gelöscht/Gelöscht/Pass-Through	Pass-Through/Gelöscht/Gelöscht
`Warn`	Pass-Through	Pass-Through
`WWW-Authenticate`	Gelöscht	Remapped

* Der `Authorization`-Header wird gelöscht, wenn er eine Signaturversion 4-Signatur enthält oder `AWS_IAM`-Autorisierung verwendet wird.

- Das Android-SDK einer von API Gateway generierten API verwendet die Klasse `java.net.HttpURLConnection`. Diese Klasse löst auf Geräten, auf denen Android 4.4 und früher ausgeführt wird, eine unbehandelte Ausnahme für eine 401-Antwort aus, die aus der Neuzuordnung des Headers `WWW-Authenticate` zu `X-Amzn-Remapped-WWW-Authenticate` resultiert.
- Im Gegensatz zu von API Gateway generierten Java-, Android- und iOS-SDKs einer API unterstützt das JavaScript SDK einer von API Gateway generierten API keine Wiederholungen bei Fehlern der Stufe 500.
- Der Test-Aufruf einer Methode verwendet den Standard-Inhaltstyp `application/json` und ignoriert Spezifikationen anderer Inhaltstypen.
- Beim Senden von Anfragen an eine API durch Übergabe des `X-HTTP-Method-Override`-Headers überschreibt API Gateway die Methode. Um den Header an das Backend zu übergeben, muss der Header der Integrationsanforderung hinzugefügt werden.
- Wenn eine Anfrage mehrere Medientypen in ihrem `Accept`-Header enthält, berücksichtigt API Gateway nur den ersten `Accept`-Medientyp. Wenn Sie die Reihenfolge der `Accept`-Medientypen nicht steuern können und der Medientyp Ihres binären Inhalts nicht der erste in der Liste ist, können Sie den ersten `Accept`-Medientyp in der Liste `binaryMediaTypes` Ihrer API hinzufügen. Amazon API Gateway gibt Ihren Inhalt dann im Binärformat zurück. Um z. B. eine JPEG-Datei mit einem `<img>`-Element in einem Browser zu übermitteln, sendet der Browser möglicherweise `Accept:image/webp,image/*,*/*;q=0.8` in einer Anforderung. Bei Hinzufügen von `image/webp` zur Liste `binaryMediaTypes` erhält der Endpunkt die JPEG-Datei als Binärdatei.
- Das Anpassen der Standard-Gateway-Antwort für `413 REQUEST_TOO_LARGE` wird derzeit nicht unterstützt.
- API Gateway enthält einen `Content-Type`-Header für alle Integrationsantworten. Der Inhaltstyp ist standardmäßig „`application/json`“.

Dokumentverlauf

Die folgende Tabelle beschreibt die wichtigen Änderungen in der Dokumentation seit der letzten Version von Amazon API Gateway. Für Benachrichtigungen über Änderungen an dieser Dokumentation können Sie einen RSS-Feed abonnieren, indem Sie auf die RSS-Schaltfläche in der oberen Menüanzeige klicken.

- Letzte Aktualisierung der Dokumentation: 15. Februar 2024

Änderung	Beschreibung	Datum
Unterstützung für TLS 1.3 hinzugefügt	API Gateway unterstützt jetzt TLS 1.3 auf regionalen REST-APIs, HTTP-APIs und WebSocket APIs. Weitere Informationen finden Sie unter Auswählen einer Sicherheitsrichtlinie für Ihre benutzerdefinierte Domain in API Gateway.	15. Februar 2024
Aktualisierungen der WebSocket REST-API und der API-Konsole	Aktualisierte Konsoleninformationen für REST-APIs und WebSocket APIs.	10. Dezember 2023
Aktualisierung der Dokumentation	Konzeptinformationen wurden aktualisiert und neue Tutorials zu den Themen „Datentransformationen“ und „Anforderungsvalidierung für API-Gateway-REST-APIs“ wurden erstellt. Weitere Informationen finden Sie unter Die Anforderungsvalidierung in API Gateway verwenden und Datentran	22. Juni 2023

	sformationen für die REST-API einrichten.	
DNS-Failover für ein API Gateway mit mehreren Regionen konfigurieren	Unterstützung für die Verwendung von Amazon Route 53-Zustandsprüfungen zur Steuerung des DNS-Failovers von einer API-Gateway-REST-API in einer primären AWS-Region zu einer in einer sekundären Region hinzugefügt. Weitere Informationen finden Sie unter Benutzerdefinierte Integritätsprüfungen für DNS-Failover konfigurieren.	31. Oktober 2022
Aktualisierung der Dokumentation	Zusammenfassungen der Kernfunktionen für REST-API- und HTTP-API-APIs wurden aktualisiert. Weitere Informationen finden Sie unter Auswählen zwischen REST-API- und HTTP-API-APIs.	31. Mai 2022
Aktualisierung der verwalteten Richtlinien	`acm:GetCertificate` - Unterstützung für `AWSServiceRoleForAPIGateway` - Richtlinien wurde hinzugefügt. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für API Gateway.	12. Juli 2021

Parameterzuordnung für HTTP-APIs	Unterstützung für die Parameterzuordnung für HTTP-APIs wurde hinzugefügt. Weitere Informationen finden Sie unter Transformieren von API-Anfragen und -Antworten.	7. Januar 2021
Standard-Endpunkt für eine REST-API deaktivieren	Unterstützung für die Deaktivierung des Standard-Endpunkts für REST-APIs wurde hinzugefügt. Weitere Informationen finden Sie unter Standard-Endpunkts für eine REST-API deaktivieren.	29. Oktober 2020
Gegenseitige TLS-Authentifizierung	Unterstützung für die gegenseitige TLS-Authentifizierung für REST-APIs und HTTP-APIs wurde hinzugefügt. Weitere Informationen finden Sie unter Konfigurieren der gegenseitigen TLS-Authentifizierung für eine REST-API und Konfigurieren der gegenseitigen TLS-Authentifizierung für eine HTTP-API.	17. September 2020
HTTP-API-Autorisierer AWS Lambda	Unterstützung für AWS Lambda Autorisierer für HTTP-APIs hinzugefügt. Weitere Informationen finden Sie unter Arbeiten mit AWS Lambda Autorisierern für HTTP-APIs.	9. September 2020

HTTP-API-Dienstintegrationen AWS	Unterstützung für AWS Serviceintegrationen für HTTP-APIs hinzugefügt. Weitere Informationen finden Sie unter Arbeiten mit AWS Serviceintegrationen für HTTP-APIs.	20. August 2020
Benutzerdefinierte HTTP-API-Platzhalterdomänen	Unterstützung für benutzerdefinierte Domänennamen mit Platzhaltern für HTTP-APIs wurde hinzugefügt. Weitere Informationen finden Sie unter Benutzerdefinierte Domänennamen mit Platzhalter.	10. August 2020
Verbesserungen des serverlosen Entwicklerportals	Benutzerverwaltung zum Administratorbereich und Support für den Export von API-Definitionen hinzugefügt. Weitere Informationen finden Sie unter Verwenden des serverlosen Entwicklerportals für die Katalogisierung Ihrer API Gateway-APIs.	25. Juni 2020
WebSocket API-Unterstützung `Sec-WebSocket-Protocol`	Unterstützung für das `Sec-WebSocket-Protocol` -Feld hinzugefügt. Weitere Informationen finden Sie unter Eine $connect-Route einrichten, für die ein WebSocket Unterprotokoll erforderlich ist.	16. Juni 2020

HTTP-API-Export	Unterstützung für den Export von OpenAPI 3.0-Definitionen von HTTP-APIs wurde hinzugefügt. Weitere Informationen finden Sie unter HTTP-API von API Gateway exportieren.	20. April 2020
Sicherheitsdokumentation	Sicherheitsdokumentation hinzugefügt. Weitere Informationen finden Sie unter Sicherheit in Amazon API Gateway.	31. März 2020
Neu strukturierte Dokumentation	Das Entwicklerhandbuch wurde neu strukturiert.	12. März 2020
Allgemeine Verfügbarkeit der HTTP-API	Veröffentlichte HTTP-APIs in der allgemeinen Verfügbarkeit. Weitere Informationen finden Sie unter Arbeiten mit HTTP-APIs.	12. März 2020
HTTP-API-Protokollierung	Unterstützung für `$context.integrationErrorMessage` in HTTP-API-Protokollen hinzugefügt. Weitere Informationen finden Sie unter HTTP-API-Protokollierungsvariablen.	26. Februar 2020
AWS Variablen für den OpenAPI-Import	Unterstützung für AWS Variablen in OpenAPI-Definitionen hinzugefügt. Weitere Informationen finden Sie unter AWS -Variablen für OpenAPI-Import.	17. Februar 2020

HTTP-APIs	Als Beta-Version veröffentlichte HTTP-APIs. Weitere Informationen finden Sie unter HTTP APIs.	4. Dezember 2019
Benutzerdefinierte Domänennamen mit Platzhalter	Unterstützung für benutzerdefinierte Domänennamen mit Platzhalter hinzugefügt. Weitere Informationen finden Sie unter Benutzerdefinierte Domänennamen mit Platzhalter.	21. Oktober 2019
Amazon Data Firehose-Protokollierung	Unterstützung für Amazon Data Firehose als Ziel für Zugriffsprotokollierungsdaten hinzugefügt. Weitere Informationen finden Sie unter Amazon Data Firehose als Ziel für die API-Gateway-Zugriffsprotokollierung verwenden.	15. Oktober 2019
Route53-Aliasnamen zum Aufrufen privater APIs	Unterstützung für zusätzliche Route53-Alias-DNS-Datensätze zum Aufrufen privater APIs hinzugefügt. Weitere Informationen finden Sie unter Aufrufen Ihrer privaten API mit Route53-Alias.	18. September 2019
Tag-basierte Zugriffskontrolle für APIs WebSocket	Unterstützung für tagbasierte Zugriffskontrolle für WebSocket APIs hinzugefügt. Weitere Informationen finden Sie unter API Gateway-Ressourcen, die markiert werden können.	27. Juni 2019

Auswahl der TLS-Version für benutzerdefinierte Domänen	Hinzufügung der Unterstützung für die Auswahl der Transport Layer Security (TLS)-Version Auswahl für APIs, die für benutzerdefinierte Domänen bereitgestellt werden. Siehe Hinweis in TLS-Mindestversion für eine benutzerdefinierte Domäne in API Gateway auswählen.	20. Juni 2019
VPC-Endpunktrichtlinien für private APIs	Hinzufügung der Unterstützung für die Verbesserung der Sicherheit privater APIs durch Anfügen von Endpunktrichtlinien an VPC-Schnittstellenendpunkte. Weitere Informationen finden Sie unter VPC-Endpunkt-Richtlinien für private APIs in API Gateway verwenden.	4. Juni 2019
Dokumentation aktualisiert	Einstieg mit Amazon API Gateway wurde neu erstellt. Tutorials nach Amazon API Gateway Tutorials verschoben.	29. Mai 2019
Tag-basierte Zugriffskontrolle für REST-APIs	Hinzufügung der Unterstützung für die Tag-basierte Zugriffskontrolle für REST APIs. Weitere Informationen finden Sie unter Tags mit IAM-Richtlinien zur Steuerung des Zugriffs auf API Gateway-Ressourcen verwenden.	23. Mai 2019

Dokumentation aktualisiert	Es wurden sechs Themen überarbeitet: Was ist Amazon API Gateway?, Tutorial: Erstellen einer API mit HTTP-Proxy-Integration, Tutorial: Erstellen einer Calc-REST-API mit drei Nicht-Proxy-Integrationen, Referenz für die API Gateway-Zuweisungsvorlage und Zugriffsprotokollierungsvariable, Verwenden von API Gateway Lambda-Genehmigern und Aktivieren von CORS für eine API Gateway-REST-API-Ressource.	5. April 2019
Verbesserungen des serverlosen Entwicklerportals	Es wurden ein Administrator-Panel und andere Verbesserungen hinzugefügt, um die Veröffentlichung von APIs im Amazon API Gateway-Entwicklerportal zu erleichtern. Weitere Informationen finden Sie unter Verwenden eines Entwicklerportals für die Katalogisierung Ihrer APIs.	28. März 2019
Support für AWS Config	Unterstützung für hinzugefügt AWS Config. Weitere Informationen finden Sie unter Überwachung der API-Gateway-API-Konfiguration mit AWS Config.	20. März 2019

Support für AWS CloudFormation	API Gateway V2 API zur AWS CloudFormation Vorlagenreferenz hinzugefügt. Weitere Informationen finden Sie unter Amazon API Gateway V2-Ressourcentypenreferenz.	7. Februar 2019
Support für WebSocket APIs	Unterstützung für WebSocket APIs hinzugefügt. Weitere Informationen finden Sie unter Erstellen einer WebSocket API in Amazon API Gateway.	18. Dezember 2018
Serverloses Entwicklerportal verfügbar über AWS Serverless Application Repository	Die serverlose Anwendung für das Amazon API Gateway Gateway-Entwicklerportal ist jetzt unter AWS Serverless Application Repository(zusätzlich zu GitHub) verfügbar. Weitere Informationen finden Sie unter Entwicklerportal zur Katalogisierung Ihrer API Gateway-APIs verwenden.	16. November 2018
Support für AWS WAF	Zusätzliche Unterstützung für AWS WAF (Web Application Firewall). Weitere Informationen finden Sie unter Steuern des Zugriffs auf eine API mithilfe von AWS WAF.	5. November 2018

Serverloses Entwicklerportal	Amazon API Gateway bietet jetzt ein vollständig anpassbares Entwicklerportal als serverlose Anwendung, das Sie zur Veröffentlichung Ihrer API Gateway-APIs einsetzen können. Weitere Informationen finden Sie unter Entwickelrportal zur Katalogisierung Ihrer API Gateway-APIs verwenden..	29. Oktober 2018
Unterstützung für mehrwertige Header- und Abfragezeichenfolgenparameter	Amazon API Gateway unterstützt jetzt mehrere gleichnamige Header und Abfragezeichenfolgenparameter. Weitere Informationen finden Sie unter Unterstützung für mehrwertige Header- und Abfragezeichenfolgenparameter.	4. Oktober 2018
OpenAPI-Support	Amazon API Gateway unterstützt jetzt sowohl OpenAPI 3.0 als auch OpenAPI (Swagger) 2.0.	27. September 2018
Dokumentation aktualisiert	Ein neues Thema wurde hinzugefügt: So wirken sich Ressourcenrichtlinien von Amazon API Gateway auf den Autorisierungs-Workflow aus.	27. September 2018

Aktive AWS X-Ray Integration	Sie können es jetzt verwenden AWS X-Ray , um Latenzen bei Benutzeranfragen zu verfolgen und zu analysieren, wenn diese über Ihre APIs zu den zugrunde liegenden Diensten weitergeleitet werden. Weitere Informationen finden Sie unter Trace API Gateway API Execution with AWS X-Ray.	6. September 2018
Caching-Verbesserungen	Nur GET-Methoden haben standardmäßig Caching aktiviert, wenn Sie das Caching für eine API-Phase aktivieren. Dies hilft, die Sicherheit Ihrer API zu gewährleisten. Sie können das Caching für andere Methoden aktivieren, indem Sie die Methodeneinstellungen überschreiben. Weitere Informationen finden Sie unter Aktivieren von API-Caching für verbesserte Reaktionsfähigkeit .	20. August 2018
Service Limits überarbeitet	Mehrere Limits wurden überarbeitet: Erhöhte Anzahl von APIs pro Konto. Erhöhte API-Ratenlimits zur Erstellung/Import/Bereitstellung von APIs. Einige Raten von pro Minute auf pro Sekunde korrigiert. Weitere Informationen finden Sie unter Limits.	13. Juli 2018

Überschreiben von API-Anforderungs- und Antwortparametern und Headern	Unterstützung für Überschreibung von Anforderungsheadern, Abfragestrings und Pfaden sowie für Antwort-Header und Statuscodes hinzugefügt. Weitere Informationen finden Sie unter Verwendung einer Zuweisungsvorlage zum Überschreiben der Anfrage- und Antwortparameter und -Header einer API.	12. Juli 2018
Drosselung auf Methodenebene für Nutzungspläne	Unterstützung für die Einrichtung von Standard-Drosel-Limits pro Methode sowie zur Einstellung von Drossel-Limits für individuelle API-Methoden in den Nutzungsplan-Einstellungen hinzugefügt. Diese Einstellungen bestehen zusätzlich zu den bestehenden Drossel-Limits auf Kontoebene und Standard-Drossel-Limits auf Methodenebene, die Sie in den Stufeneinstellungen festlegen können. Weitere Informationen finden Sie unter Drosselungs-API-Anfragen für einen besseren Durchsatz.	11. Juli 2018

API Gateway Entwicklerhandbuch Update-Benachrichtigungen jetzt über RSS verfügbar	Die HTML-Version des API Gateway Entwicklerhandbuchs unterstützt jetzt einen RSS-Feed mit Aktualisierungen, die auf dieser Dokumentenhistorie-Seite dokumentiert sind. Der RSS-Feed umfasst Aktualisierungen ab 27. Juni 2018 und später. Zuvor angekündigte Updates stehen weiterhin auf dieser Seite zur Verfügung. Verwenden Sie die RSS-Schaltfläche in der oberen Menüanzeige, um den Feed zu abonnieren.	27. Juni 2018

Frühere Updates

Die folgende Tabelle beschreibt wichtige Änderungen in jeder Version des API Gateway Entwicklerhandbuchs vor dem 27. Juni 2018.

Änderungen	Beschreibung	Datum geändert
Private APIs	Unterstützung für private APIs hinzugefügt, die Sie über Schnittstellen-VPC-Endpunkte verfügbar machen. Der Datenverkehr zu Ihren privaten APIs verlässt das Amazon-Netzwerk nicht. Er ist vom öffentlichen Internet isoliert.	14. Juni 2018
Kontoübergreifende Lambda-Autorisierungen und -Integrationen und kontoübergreifende Amazon Cognito-B	Verwenden Sie eine AWS Lambda Funktion aus einem anderen AWS Konto als Lambda-Autorisierungsfunktion oder als API-Integrations-Backend. Oder verwenden Sie einen Amazon Cognito-Benutzerpool als Autorisierungsfunktion. Das andere Konto kann sich in jeder Region befinden, in der Amazon API Gateway verfügbar ist. Weitere Informationen finden Sie unter the section called	2. April 2018

Änderungen	Beschreibung	Datum geändert
enutzerpool-Autorisierungen	“Kontenübergreifenden Lambda-Genehmiger konfigurieren”, the section called “Tutorial: API mit kontenübergreifender Lambda-Proxy-Integration erstellen” und the section called “Kontenübergreifenden Amazon Cognito-Genehmiger für eine REST-API konfigurieren”.	
Ressourcenrichtlinien für APIs	Mit Amazon-API-Gateway-Ressourcenrichtlinien ermöglichen Sie Benutzern eines anderen AWS -Kontos, sicher auf Ihre API zuzugreifen oder legen fest, dass die API nur von einem bestimmten IP-Adressbereich oder bestimmten CIDR-Blöcken aufgerufen werden kann. Weitere Informationen finden Sie unter the section called “API Gateway-Ressourcenrichtlinien verwenden”.	2. April 2018
Markieren bei API Gateway-Ressourcen	Markieren Sie eine API-Stufe mit bis zu 50 Tags für die Kostenzuweisung von API-Anfragen und das Caching in API Gateway. Weitere Informationen finden Sie unter the section called “Einrichten von Tags”.	19. Dezember 2017
Nutzlastkomprimierung und -dekomprimierung	Aktivieren Sie den Aufruf Ihrer API mit komprimierten Nutzlasten mithilfe einer der unterstützten Inhaltscodierungen. Die komprimierten Nutzlasten werden zugeordnet, wenn eine Textzuweisungsvorlage angegeben wird. Weitere Informationen finden Sie unter the section called “Inhaltskodierung”.	19. Dezember 2017
Von einem Genehmiger stammender API-Schlüssel	Geben Sie einen API-Schlüssel von einem benutzerdefinierten Genehmiger an API Gateway zurück, um einen Nutzungsplan für API-Methoden anzuwenden, die den Schlüssel erfordern. Weitere Informationen finden Sie unter the section called “Auswählen einer API-Schlüsselquelle”.	19. Dezember 2017

Änderungen	Beschreibung	Datum geändert
Autorisierung mit OAuth-2-Bereichen	Aktivieren Sie die Autorisierung des Methodenaufrufs durch Verwendung von OAuth-2-Bereichen mit dem `COGNITO_USER_POOLS` -Genehmiger. Weitere Informationen finden Sie unter the section called “Amazon Cognito-Benutzerpool als Genehmiger für eine REST-API verwenden”.	14. Dezember 2017
Private Integration und VPC-Link	Erstellen Sie eine API mit der privaten API Gateway Gateway-Integration, um Kunden den Zugriff auf HTTP/HTTPS-Ressourcen in einer Amazon-VPC von außerhalb der VPC über eine Ressource zu ermöglichen. VpcLink Weitere Informationen erhalten Sie unter the section called “Tutorial: Erstellen einer API mit privater Integration” und the section called “Private Integration”.	30. November 2017
Bereitstellen eines Canary für API-Tests	Fügen Sie einer vorhandenen API-Bereitstellung ein Canary-Release hinzu, um eine neuere Version der API zu testen, während die aktuelle Version auf derselben Stufe in Betrieb ist. Sie können einen Prozentsatz des Stage-Traffics für die Canary-Version festlegen und die Ausführung und den Zugriff auf bestimmte Kanaren aktivieren, die in separaten Log-Logs protokolliert werden. CloudWatch Weitere Informationen finden Sie unter the section called “Einrichten einer Canary-Release-Bereitstellung”.	28. November 2017
Zugriffsprotokollierung	Protokollieren Sie den Client-Zugriff auf Ihre API mit Daten aus $context-Variablen in einem Format Ihrer Wahl. Weitere Informationen finden Sie unter the section called “CloudWatch Logs”.	21. November 2017

Änderungen	Beschreibung	Datum geändert
Ruby-SDK auf einer API	Erstellen Sie ein Ruby-SDK für Ihre API und nutzen Sie diesen für den Aufruf Ihrer API-Methoden. Weitere Informationen erhalten Sie unter the section called “Generieren des Ruby SDK einer API” und the section called “Ein von API Gateway generiertes Ruby-SDK für eine REST-API verwenden”.	20. November 2017
Regionaler API-Endpunkt	Geben Sie einen regionalen API-Endpunkt an, um eine API für nicht mobile Clients zu erstellen. Ein nicht mobiler Client wie eine EC2-Instance wird in derselben AWS -Region wie die bereitgestellte API ausgeführt. Wie bei einer Edge-optimierten API können Sie einen benutzerdefinierten Domänennamen für eine regionale API erstellen. Weitere Informationen erhalten Sie unter the section called “API Gateway Gateway-Endpunkttypen” und the section called “Einrichten eines regionalen benutzerdefinierten Domänennamens”.	2. November 2017
Benutzerdefinierter Genehmiger einer Anforderung	Verwenden Sie den benutzerdefinierten Genehmiger für eine Anforderung, um Benutzer-Authentifizierungsinformationen in Anforderungparametern bereitzustellen und API-Methodenaufrufe zu genehmigen. Zu den Anforderungparametern gehören Header- und Abfragezeichenfolgenparameter, ebenso wie Stufen- und Kontextvariablen. Weitere Informationen finden Sie unter API Gateway-Lambda-Genehmiger verwenden.	15. September 2017

Änderungen	Beschreibung	Datum geändert
Anpassen von Gateway-Antworten	Anpassen von API Gateway-generierten Gateway-Antworten auf API-Anfragen, die das Integrations-Backend nicht erreichten. Eine angepasste Gateway-Nachricht kann API-spezifische benutzerdefinierte Fehlermeldungen für den Aufrufer enthalten – darunter die erforderlichen CORS-Header – oder die Gateway-Antwortdaten in ein externes Austauschformat umwandeln. Weitere Informationen finden Sie unter Einrichten von Gateway-Antworten, um Fehlerantworten anzupassen.	6. Juni 2017
Benutzerdefinierte Lambda-Fehlereigenschaften auf Methoden-Antwortheader zuordnen	Ordnen Sie einzelne benutzerdefinierte Fehlereigenschaften, die von Lambda zurückgegeben werden, den Antwort-Header-Parametern der Methode mit dem `integration.response.body` -Parameter zu, wobei API Gateway zur Deserialisierung des benutzerdefinierten Zeichenfolgen-Fehlerobjekts zur Laufzeit verwendet wird. Weitere Informationen finden Sie unter Benutzerdefinierte Lambda-Fehlern in API Gateway behandeln.	6. Juni 2017
Erhöhen der Drosselungslimits	Erhöhen Sie die Begrenzung der konstanten Anforderungsrate auf Kontoebene auf 10 000 Anforderungen pro Sekunde (RPS) und die Burst-Begrenzung auf 5 000 gleichzeitige Anforderungen. Weitere Informationen finden Sie unter Drosseln der API-Anforderungen für einen besseren Durchsatz.	6. Juni 2017
Validieren von Methodenanforderungen	Konfigurieren Sie grundlegende Anfragevalidierer auf API- oder Methodenebene, sodass API Gateway eingehende Anfragen validieren kann. Amazon API Gateway überprüft, ob die erforderlichen Parameter festgelegt und nicht leer sind und ob das Format der anwendbaren Nutzlasten mit dem konfigurierten Modell konform ist. Weitere Informationen finden Sie unter Verwenden der Anforderungsvalidierung in API Gateway.	11. April 2017

Änderungen	Beschreibung	Datum geändert
Integration in ACM	Verwenden Sie ACM-Zertifikate für die benutzerdefinierten Domänennamen Ihrer API. Sie können ein Zertifikat in ACM erstellen AWS Certificate Manager oder ein vorhandenes Zertifikat im PEM-Format in ACM importieren. Wenn Sie einen benutzerdefinierten Domänennamen für Ihre APIs festlegen, verwenden Sie den ARN des Zertifikats. Weitere Informationen finden Sie unter Einrichten von benutzerdefinierten Domänennamen für REST-APIs.	9. März 2017
Generieren und Aufrufen des Java-SDKs einer API	Lassen Sie API Gateway das Java-SDK für Ihre API generieren und verwenden Sie das SDK, um die API in Ihrem Java-Client aufzurufen. Weitere Informationen finden Sie unter Von API Gateway generiertes Java-SDK für eine REST-API verwenden.	13. Januar 2017
Integrieren mit AWS Marketplace	Verkaufen Sie Ihre API in einem Nutzungsplan als SaaS-Produkt über AWS Marketplace. und erhöhen Sie mit AWS Marketplace die Reichweite für Ihre API. Verlassen Sie sich AWS Marketplace auf die Kundenabrechnung in Ihrem Namen. Überlassen Sie API Gateway die Benutzerautorisierung und Nutzungsmessung. Weitere Informationen finden Sie unter Verkaufen Ihrer APIs als SaaS.	1. Dezember 2016
Aktivieren der Dokumentationsunterstützung für Ihre API	Fügen Sie Dokumentation für API-Entitäten in DocumentationPartRessourcen in API Gateway hinzu. Ordnen Sie einen Snapshot der `DocumentationPart` Sammlungsinstanzen einer API-Phase zu, um eine zu erstellen DocumentationVersion. Veröffentlichen Sie die API-Dokumentation durch Exportieren einer Dokumentationsversion in eine externe Datei, beispielsweise eine OpenAPI-Datei. Weitere Informationen finden Sie unter Dokumentieren von REST-APIs.	1. Dezember 2016

Änderungen	Beschreibung	Datum geändert
Aktualisierung des benutzerdefinierten Genehmigers	Eine Kunden-Genehmiger-Lambda-Funktion gibt jetzt die Prinzipal-ID des Aufrufers zurück. Die Funktion kann auch andere Informationen zurückgeben, darunter Schlüssel-Wert-Paare der `context`-Zuordnung und eine IAM-Richt linie. Weitere Informationen finden Sie unter Ausgabe von einem API Gateway Lambda Authorizer.	1. Dezember 2016
Unterstützung binärer Nutzlasten	Stellen binaryMediaTypesSie Ihre API so ein, dass binäre Payloads einer Anfrage oder Antwort unterstützt werden. Legen Sie die `contentHandling` Eigenschaft für eine Integration fest oder geben Sie IntegrationResponsean, ob eine binäre Nutzlast als systemeigener Binär-Blob, als Base64-kodierte Zeichenfolge oder als Passthrough ohne Änderungen behandelt werden soll. Weitere Informationen finden Sie unter Mit binären Medientypen für REST-APIs arbeiten.	17. November 2016
Aktivieren einer Proxy-Integration mit einem HTTP- oder Lambda-Ba ckend über die Proxy-Ressource einer API.	Erstellen Sie eine Proxy-Ressource mit einem Greedy-Pa th-Parameter der Form `{proxy+}` und der Catch-all `ANY`-Methode. Die Proxy-Ressource wird mit Hilfe der HTTP- bzw. Lambda-Proxy-Integration in ein HTTP- oder Lambda-Backend integriert. Weitere Informationen finden Sie unter Einrichten der Proxy-Integration mit einer Proxy-Ressource.	20. September 2016
Erweitern Sie ausgewählte APIs in API Gateway als Produktangebot für Ihre Kunden durch Bereitstellung eines oder mehrerer Nutzungspläne.	Erstellen Sie einen Nutzungsplan in API Gateway, um ausgewählten API-Kunden den Zugriff auf bestimmte API-Stufen zu vereinbarten Anfrageraten und Quoten zu ermöglichen. Weitere Informationen finden Sie unter Erstellen und Verwenden von Nutzungsplänen mit API-Schlüsseln.	11. August 2016

Änderungen	Beschreibung	Datum geändert
Autorisierung auf Methodenebene mit einem Benutzerpool in Amazon Cognito ermöglichen	Erstellen Sie einen Benutzerpool in Amazon Cognito und verwenden Sie ihn als Ihren eigenen Identitätsanbieter . Sie können den Benutzerpool für die Autorisierung auf Methodenebene konfigurieren, um Benutzern, die im Benutzerpool registriert sind, Zugriff zu gewähren. Weitere Informationen finden Sie unter Zugriff auf eine REST-API mit Amazon Cognito-Benutzerpools als Genehmiger steuern.	28. Juli 2016
Aktivierung von CloudWatch Amazon-Metriken und -Dimensionen unter dem `AWS/ApiGateway` Namespace.	Die API Gateway Gateway-Metriken sind jetzt unter dem CloudWatch Namespace von `AWS/ApiGateway` standardisiert. Sie können sie sowohl in der API Gateway Gateway-Konsole als auch in der CloudWatch Amazon-Konsole anzeigen. Weitere Informationen finden Sie unter Amazon API Gateway-Dimensionen und -Metriken.	28. Juli 2016
Aktivieren der Zertifikatrotation für einen benutzerdefinierten Domänennamen	Mit der Zertifikatrotation können Sie ein ablaufendes Zertifikat für einen benutzerdefinierten Domänennamen hochladen und erneuern. Weitere Informationen finden Sie unter Wechseln eines in ACM importierten Zertifikats.	27. April 2016
Dokumentationsänderungen für die aktualisierte Amazon API Gateway-Konsole	Erfahren Sie, wie Sie eine API mit der aktualisierten API Gateway-Konsole erstellen und einrichten. Weitere Informationen erhalten Sie unter Tutorial: Erstellen einer REST-API durch Importieren eines Beispiels und Tutorial: REST-API mit HTTP-API ohne Proxy-Integration erstellen.	5. April 2016

Änderungen	Beschreibung	Datum geändert
Aktivieren der Funktion zum Importieren von APIs, um aus externen API-Definitionen neue APIs zu erstellen oder vorhandene zu aktualisieren	Mit der Funktion zum Importieren von APIs erstellen Sie eine neue API oder aktualisieren eine vorhandene, indem Sie mit den API Gateway-Erweiterungen eine externe, in Swagger 2.0 angegebene API-Definition hochladen. Weitere Informationen über die Funktion zum Importieren von APIs erhalten Sie unter Konfigurieren einer REST-API mit OpenAPI.	5. April 2016
Offenlegen der `$input.body` - Variablen, um auf die rohe Nutzlast als Zeichenfolge zuzugreifen, und der `$util.parseJson()` - Funktion, um eine JSON-Zeichenfolge in ein JSON-Objekt in einer Zuweisungsvorlage zu verwandeln	Weitere Informationen zu `$input.body` und `$util.parseJson()` finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen.	5. April 2016
Aktivieren von Client-Anforderungen mit Cache-Invalidierung auf Methodenebene und Verbesserung der Verwaltung der Anforderungsdrosselung	Leeren Sie den Cache auf der Ebene der API-Stufen oder heben Sie die Gültigkeit einzelner Cache-Einträge auf. Weitere Informationen erhalten Sie unter API-Stufen-Cache in API Gateway leeren und API Gateway-Cache-Eintrag ungültig machen. Verbessern Sie die Konsolenumgebung für die Verwaltung der Drosselung von API-Anforderungen. Weitere Informationen finden Sie unter Drosseln der API-Anforderungen für einen besseren Durchsatz.	25. März 2016

Änderungen	Beschreibung	Datum geändert
API Gateway-API mithilfe von benutzerdefinierter Autorisierung aktivieren und aufrufen	Erstellen und konfigurieren Sie eine AWS Lambda Funktion zur Implementierung einer benutzerdefinierten Autorisierung. Die Funktion gibt ein IAM-Richtliniendokument zurück, das die Allow- oder Deny-Berechtigungen für Client-Anfragen einer API Gateway-API gewährt. Weitere Informationen finden Sie unter API Gateway-Lambda-Genehmiger verwenden.	11. Februar 2016
API Gateway-API unter Verwendung einer Swagger-Definitionsdatei und Erweiterungen importieren und exportieren	Erstellen und aktualisieren Sie Ihr API Gateway mithilfe der Swagger-Spezifikation mit den API Gateway-Erweiterungen. Importieren Sie die OpenAPI-Definitionen mit der API Gateway-Importfunktion. Exportieren Sie eine API Gateway-API in eine Swagger-Definitionsdatei unter Verwendung der API Gateway-Konsole oder der API Gateway Export-API. Weitere Informationen erhalten Sie unter Konfigurieren einer REST-API mit OpenAPI und REST-API von API Gateway importieren.	18. Dezember 2015
Zuordnung von Anforderungs- oder Antworttext bzw. JSON-Feldern im Text zu Anforderungs- oder Antwortparametern	Weisen Sie den Methodenanforderungstext oder dessen JSON-Felder dem Pfad, der Abfragezeichenfolge oder den Headern der Integrationsanforderung zu. Ordnen Sie den Integrationsantworttext oder dessen JSON-Felder den Antwort-Headern der Anforderung zu. Weitere Informationen finden Sie unter Daten-Mapping-Referenz für Amazon API Gateway-API-Anfragen und Antworten.	18. Dezember 2015
Mit Stage-Variablen in Amazon API Gateway arbeiten	Erfahren Sie, wie Konfigurationsattribute mit einer Bereitstellungsphase einer API in Amazon API Gateway verknüpft werden können. Weitere Informationen finden Sie unter Einrichten von Stufenvariablen für eine REST-API-Bereitstellung.	5. November 2015

Änderungen	Beschreibung	Datum geändert
Vorgehensweise: Aktivieren von CORS für eine Methode	Es ist jetzt einfacher, Cross-Origin Resource Sharing (CORS) für Methoden in Amazon API Gateway zu aktiviere n. Weitere Informationen finden Sie unter Aktivieren von CORS für eine REST-API-Ressource.	3. November 2015
Vorgehensweise: Verwenden von clientseitiger SSL-Authentifizierung	Verwenden Sie Amazon API Gateway zur Generieru ng von SSL-Zertifikaten, die Sie zur Authentifizierung von Aufrufen an Ihr HTTP-Backend verwenden können. Weitere Informationen finden Sie unter Erstellen und Konfigurieren eines SSL-Zertifikats für die Backend-A uthentifizierung.	22. September 2015
Pseudointegration von Methoden	Erfahren Sie, wie die Mock-Integration einer API mit Amazon API Gateway funktioniert. Diese Funktion ermöglicht es Entwicklern, API-Antworten direkt über API Gateway zu generieren, ohne dass zuvor ein endgültiges Integrations-Backend erforderlich ist.	1. September 2015
Unterstützung von Amazon Cognito Identity	Amazon API Gateway hat den Bereich der `$context`-Variablen erweitert, sodass jetzt Informationen über Amazon Cognito Identity zurückgegeben werden, wenn Anfragen mit Amazon Cognito-Anmeldeinformationen signiert werden. Darüber hinaus haben wir eine `$util` Variable hinzugefügt, mit der Zeichen in URLs und Zeichenketten JavaScript maskiert und kodiert werden können. Weitere Informationen finden Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokoll ierungsvariablen.	28. August 2015

Änderungen	Beschreibung	Datum geändert
OpenAPI-Integration	Verwenden Sie das Swagger-Importtool auf GitHub, um Swagger-API-Definitionen in Amazon API Gateway zu importieren. Erfahren Sie mehr über die API Gateway-Erweiterungen für OpenAPI arbeiten, um APIs und Methoden mit dem Import-Tool zu erstellen und bereitzustellen. Mit dem OpenAPI-Import-Tool können Sie auch vorhandene APIs aktualisieren.	21. Juli 2015
Referenz für die Zuweisungsvorlage	Erfahren Sie unter Referenz zu API Gateway-Mapping-Vorlage und -Zugriffsprotokollierungsvariablen mehr über den `$input`-Parameter und seine Funktionen.	18. Juli 2015
Erste veröffentlichte Version	Dies ist die erste öffentliche Version des API Gateway-Entwicklerhandbuch.	9. Juli 2015

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

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.